基于仿真模型的动态响应优化方法

毛虎平　著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书系统地介绍了机械结构动态响应优化方法，包括并行优化、基于时间谱元法的机械系统物理模型求解、多元模型自适应与时间谱元法结合的动态响应优化技术、基于 MARS 的动态响应优化算法、基于模糊聚类的全局动态响应优化算法等。

本书内容较为丰富，具有较强的前沿性和可操作性，可供从事机械系统或结构动态响应分析和优化设计方面的工程师参考，也可作为研究生或高年级本科生动态响应优化课程的参考教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

基于仿真模型的动态响应优化方法/毛虎平著. —北京：电子工业出版社，2014.3
ISBN 978-7-121-18285-3

Ⅰ. ①基… Ⅱ. ①毛… Ⅲ. ①动态响应－最优化算法 Ⅳ. ①O32

中国版本图书馆 CIP 数据核字（2014）第 037876 号

策划编辑：秦绪军　赵　娜
责任编辑：谭丽莎
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：720×1000　1/16　印张：14.25　字数：296 千字
印　　次：2014 年 3 月第 1 次印刷
定　　价：42.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

机械结构几乎都在动态载荷的环境下工作，其各种性能都是依赖于时间的函数。为了提升机器的动态性能，动态优化是非常必要的，然而当前动态优化设计的理论与方法难以适应现代化产品设计的需要，这点主要表现在以下几个方面：首先，传统的机械结构优化几乎都是静态优化，缺乏考虑由于动态载荷的作用而产生的动态效应，即在静态力作用下，应用经典优化算法优化，难以解决动态载荷作用下机器性能最佳的问题；其次，即便进行动态优化，也只是直接进行优化，从而造成因动态分析的复杂性和高耗时性而产生极慢收敛，甚至发散的现象。鉴于此，中北大学能源与动力工程专业从 2007 年将本书作者派往华中科技大学攻读博士学位，师从国内外动态仿真优化专家陈立平教授和吴义忠教授，重点研究结构在动态载荷作用下的优化理论与方法。

结构动态响应优化是为实现机械系统或机械结构在动态载荷作用下，其性能或结构动态响应达到最佳或最大响应最小，其目标是实现对机械系统单一或组合的物理模型，或者结构有限元模型进行快速、高效和准确的最优设计。与传统的动态优化不同的是，本书作者从并行优化、机械系统物理模型求解，以及处理与时间相关的约束三个方面出发，并考虑动态响应优化本质，系统地研究了结构动态响应优化，从而快速、高效和准确地实现了机械系统或结构动态响应优化。

产品的动态响应优化是产品建模和仿真的最终目的。动态响应优化是指基于机械系统或结构的仿真的参数优化，对机械系统物理模型或结构的有限元模型，采用相关的优化搜索算法进行求解的一整套方法。基于仿真的动态响应优化最重要的特点是在优化迭代过程中需要通过仿真求解来完成目标函数和约束的估值，而传统的优化算法或启发式算法因需要大量的仿真估值而显得力不从心，加上动态响应仿真分析的复杂性和高耗时性，使得传统的优化方法雪上加霜。

本书章节安排如下：第 1 章对基于仿真的动态响应优化方法及其相关技术进行概述；第 2 章从优化并行化的角度考虑动态响应优化问题；第 3 章介绍基于时间谱元法的动态响应优化方法，包括机械系统物理模型的建立，时间谱元法的模型转化，以及处理与时间相关的约束等；第 4 章介绍多元模型自适应与时间谱元法结合的动态响应优化，包括多元模型自适应优化算法原理、算法实现、如何与时间谱元法混合，以及时间关键点集方法处理与时间相关的约束；第 5 章介绍基于 MARS 的动态响应优化算法，包括 MARS 响应面的建立、移动极限策略，以

及两者的混合、置信域方法；第 6 章介绍基于模糊聚类的全局动态响应优化算法，包括模糊聚类理论、Kriging 元模型的建立及全局动态响应搜索的方法。

本书涉及的研究工作得到了国家自然基金项目“基于等效体积应变静态载荷的解空间谱元离散关键点方法及其在结构动态响应优化中的应用”（资助号：51275489）、山西省基础研究计划——煤层气联合研究基金项目（资助号：2012012003）和煤与煤层气共采山西省重点实验室项目资助，在此表示诚挚的谢意。同时，对为本书研究提出许多宝贵意见的华中科技大学 CAD 中心的陈立平教授和吴义忠教授，CAD 中心的陶松桥博士（现在武汉交通职业学院）和魏昕博士（现在武汉轻工大学机械工程学院），以及中北大学机电工程学院的郭保全副教授，中北大学理学院的李建军副教授，中北大学能源与动力工程的在读博士张艳岗、王悦芳，在读硕士魏波、王杰等，在此一并表示衷心的感谢！

动态响应优化是一个较新的研究领域，且仍处于蓬勃的发展阶段，加上作者的水平有限，本书难免存在疏漏之处，欢迎读者批评指正。

作　　者

目　录

第 1 章　绪论 …… 1

1.1　研究目的和意义 …… 3
1.2　国内外研究概况 …… 5
1.2.1　动态响应优化 …… 6
1.2.2　基于仿真的优化研究概况 …… 7
1.3　本书研究内容、创新点与组织结构 …… 13
1.3.1　本书的主要研究内容 …… 13
1.3.2　主要创新点 …… 14
1.3.3　组织结构 …… 15

第 2 章　动态响应优化问题的并行化处理 …… 17

2.1　动态响应优化算法的并行化 …… 17
2.1.1　SQP 算法的并行化 …… 17
2.1.2　并行调度算法 …… 18
2.1.3　算法分析 …… 22
2.2　动态响应并行优化过程 …… 23
2.2.1　并行平台 …… 23
2.2.2　动态响应并行优化实现 …… 24
2.3　实例分析 …… 27
2.4　本章小结 …… 32

第 3 章　基于时间谱元法的动态响应优化算法 …… 33

3.1　谱元法 …… 33
3.1.1　瞬态和稳态分析 …… 34
3.1.2　全部状态变量的全局组装和求解 …… 36
3.2　逐步时间谱元法 …… 37
3.2.1　线性动力学方程的逐步时间谱元法 …… 38
3.2.2　线性结构动态响应方程及其转化形式 …… 38
3.2.3　时间分段及单元划分 …… 39

3.2.4 单元分析 39
3.2.5 集成总体时间谱元方程及求解 41
3.3 初始条件 41
3.4 基于时间谱元法的动态响应优化 41
3.5 实例分析 42
3.5.1 动态响应分析 42
3.5.2 动态响应优化 47
3.6 本章小结 83
第 4 章 多元模型自适应与时间谱元法结合的动态响应优化 84
4.1 时间谱元法 86
4.1.1 结构动力学方程及其转化形式 87
4.1.2 单元划分 87
4.1.3 单元分析 88
4.1.4 集成总体时间谱元方程并求解 88
4.2 元模型混合自适应方法 89
4.3 处理与时间相关约束的方法 90
4.3.1 关键点集法（Critical Points Set Method，CPSM） 90
4.3.2 元模型混合自适应优化依赖于时间的约束 91
4.4 算例分析 92
4.4.1 汽车悬架系统动态响应优化设计问题一 92
4.4.2 汽车悬架系统动态响应优化设计问题二 94
4.5 结论 96
4.6 本章小结 97
第 5 章 基于 MARS 的动态响应优化算法 98
5.1 响应面方法 98
5.1.1 多项式 98
5.1.2 MARS 99
5.2 基于 MARS 的优化技术 103
5.2.1 优化问题 103
5.2.2 近似概念 103
5.2.3 基于 MARS 的动态优化策略 104
5.3 实例分析 113
5.3.1 Hesse 函数 113
5.3.2 阶梯悬臂梁截面设计 113

5.3.3　变截面悬臂梁形状优化设计……116
5.3.4　线性两自由度减振器最优设计……120
5.3.5　汽车悬架系统动态响应优化设计……124
5.4　本章小结……130
第 6 章　基于模糊聚类的全局动态响应优化算法……131
6.1　模糊数学方法……131
6.1.1　模糊集的概念……132
6.1.2　模糊分类关系……133
6.1.3　模糊聚类……134
6.2　模糊 C 均值聚类……136
6.3　模糊聚类全局仿真优化算法……137
6.3.1　Kriging 元模型……137
6.3.2　拉丁超立方试验设计……140
6.3.3　算法思想及步骤……140
6.4　实例分析……144
6.5　本章小结……154
附录 A　动态响应优化问题的并行化处理程序……155
附录 B　基于时间谱元法的动态响应优化算法程序……159
附录 C　多元模型自适应与时间谱元法结合的动态优化……177
附录 D　MARS 的动态响应优化算法……187
附录 E　基于模糊聚类的全局动态响应优化算法……200
参考文献……210

第1章 绪 论

产品建模与仿真的最终目的是实现产品的优化设计。仿真优化是指基于系统仿真的参数优化，它是针对仿真模型建立优化问题，并采用相关的优化搜索算法进行求解的一整套技术，是基于仿真的目标和约束的优化问题，其原理如图 1.1 所示，即基于模型仿真给出的输入关系构造优化模型，通过输出给优化算法得到最佳的输入量。许多工程实际问题都能归结为基于仿真的优化问题，如制造系统、交通系统、电力系统、化工系统等[1~3]。而基于仿真模型的动态响应优化不仅是仿真优化问题，更是动态响应优化问题。从狭义上讲，动态响应优化是在动载荷作用下的结构动态特性的最优化；从广义上讲，动态响应优化是优化目标函数或约束函数与时间相关，而设计变量与时间无关的优化，当然后者包含前者。

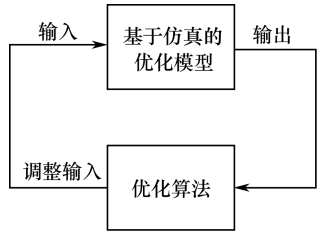


图 1.1 基于仿真模型的优化原理

基于仿真模型的动态响应优化的特点总结如下。

（1）系统的输入/输出关系有两种情况：一种是缺少结构信息，不存在解析表达式，仅能通过仿真得到近似解；另一种是虽然存在解析表达式，但是获得解析解比较困难，如微分方程或偏微分方程等，或者解析表达式采用近似方法获得，如结构的运动模型。

（2）不论是哪一种仿真模型都存在不确定因素，仿真一次只能得到对应某一输入的一次性能估计，一般都存在误差。

（3）在基于仿真的优化中，一种仿真耗费时间较少；另一种仿真耗费时间多，且缺少针对耗费时间多的仿真优化的算法，导致优化过程十分耗时，使人无法容忍甚至是不可能实现的。

（4）仿真模型与时间有关，这样中间变量多，约束条件多，优化的目标不在一个点而是在一条超曲线或一个超曲面上，因此要在所有时间点上都满足约束，这样处理约束极为困难；并且当存在多个极小值时，很难实现全局最优。

如果最优化问题的解不随时间而变，则称为静态最优化（参数最优化）问题，如变截面梁的外形形状设计问题、稳态电网的最优设计问题、阶梯梁截面尺寸设计问题和电机及变压器的最优设计问题等；如果最优化问题的解随时间而变，即状态变量是时间 t 的函数，则是动态响应最优化问题，如桁架结构在动态载荷作

用下的优化设计（如图 1.2 所示）、减振器设计问题（如图 1.3 所示）、汽车悬架系统设计问题、连杆在动态载荷作用下的响应优化问题（如图 1.4 所示）和曲柄滑块机构的动态响应优化设计（如图 1.5 所示）等，在这种情况下，问题的约束要在每一个时间点都满足，如何处理约束是决定优化是否成功的关键；如果在求解最优化问题中仿真耗时较多，则称为计算昂贵的仿真动态响应优化问题，在这种情况下，减少仿真次数显得尤为重要。仿真优化最重要的特点是在优化迭代过程

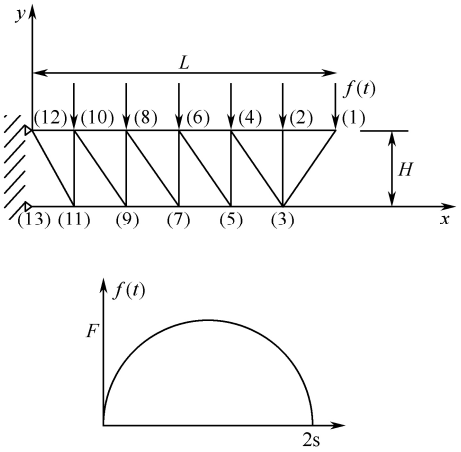


图 1.2　桁架结构和动态载荷

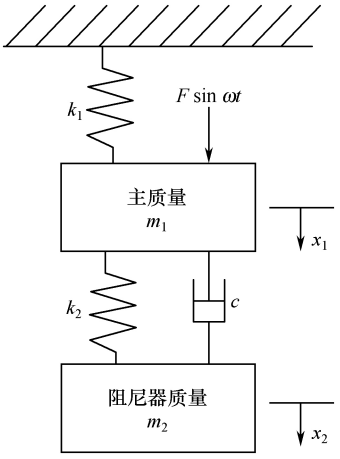


图 1.3　减振器设计问题

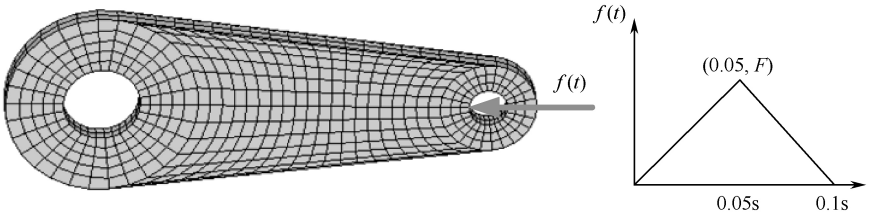


图 1.4　连杆动态响应优化

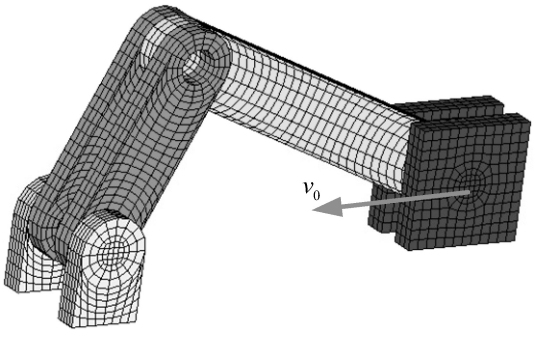


图 1.5　曲柄滑块机构

中需要通过仿真求解来完成目标函数和约束函数的估值，因此无论是解动态响应优化问题还是计算昂贵的仿真动态响应优化问题，传统的优化算法或启发式算法因为需要大量的仿真估值而显得力不从心。

鉴于基于仿真模型的动态响应优化的工程背景和上述难点，它一直是各领域学者和工程师们共同关注的重要课题，尤其是在机械制造和航空航天等领域。随着计算机技术、人工智能和数学分析方法的发展，基于仿真模型的动态响应优化的研究更加显得迫在眉睫。

1.1　研究目的和意义

本书是在确定性优化和随机优化的基础上进行研究的，其目的有如下几点。

1. 在优化库上实现序列二次规划法的仿真动态响应优化的并行化

并行优化可以在一定程度上提高仿真优化的效率，从问题分解的角度解决基于仿真的动态响应优化耗时问题。约束变尺度法又被称为序列二次规划（SQP）算法，具有收敛快、效率高、可靠性与整体收敛性好、适应能力强等一系列优点[4]，并被成功应用于工程优化问题的求解。序列二次规划算法通过求解一系列的二次规划子问题（QP）来获得原问题的最优解，而其二次规划子问题的精度依赖于原问题的目标函数和约束函数梯度值的精度，并对问题的收敛性有较大影响。对于基于多学科仿真的多实例（即多工况或一个系列产品）、多目标仿真优化问题，当采用差分法计算梯度时，由于每次迭代需要进行多次仿真函数评估（评估次数等于设计变量维数的两倍乘以实例数+1），使得优化求解过程很费时。应用 SQP 优化的并行优化可以解决动态响应优化耗时多的问题。

2. 精确求解机械系统动态响应，基于 GLL 点建立近似模型，提出在机械动态响应中，解决所有时间点都必须满足约束的最佳优化方案

机械结构几乎都在动态载荷作用下工作，其性能均是时间的函数。为了提升机器的性能，进行动态响应优化显得尤为重要。对于动态响应优化而言，首先要分析载荷在时间域上的变化，然后求出精确的离散点的响应，并满足与时间相关的约束；为了防止约束在任意两个时间离散点之间失效，可采用小时间步来计算响应。然而，优化器在每一步迭代中，需要重新计算目标函数、约束函数和灵敏度。因此，本书作者采用时间谱元法精确高效地求解响应。

本书采用两种处理与时间相关约束的方法：一种是 GLL 点法；另一种是关键点法。第一种方法将时间离散为尽量多的 GLL 点，因此约束数量较多，且计算量比较大；第二种方法将时间离散为尽量少的 GLL 点，对每一个单元进行高次

Lagrange 插值，然后通过一维搜索找到单元的极值点，因此约束数量比较少，计算开销小。

3. 利用 MARS 的特点，结合移动极限策略和置信域方法及数据驱动研究适合于基于计算昂贵的黑箱仿真模型的动态响应优化算法

MARS 是一个自适应的回归过程，适合于解决多维问题。它采用了将高维问题简化为低维高精度模型的被修改的回归分块策略。MLS 是在设计空间确定子区间的位置和大小的方法，它不仅反映了函数近似质量，而且反映了优化过程的收敛历史。其目的是减少传统响应面的不利因素，特别是对于高维非线性问题而言。

数据驱动仿真动态响应优化方法，将增广拉格朗日方法、有效的数值处理技术和多变量自适应回归样条有机地结合了起来。在本方法中，只有当最优点在数据库中，或者有很近的点在数据库中，才用数据库中的点对应的目标函数和约束函数的值代替最优点的目标函数和约束函数的值，而不需要再进行计算昂贵的仿真分析，如果不存在这样的点，只能通过计算昂贵的仿真分析获得最优点的目标函数和约束函数的值。这样随着优化的进行，数据库越来越成熟了。将局部 MARS 响应面、数据处理技术与增广拉格朗日结合起来可以减少计算昂贵的仿真分析次数。

4. 研究基于模糊聚类和近似模型的全局动态响应优化方法，目的是减少计算昂贵的仿真评估次数

对于计算昂贵的仿真优化问题，研究的重点在于减少仿真函数的评估和计算昂贵分析的次数。这是因为在优化过程中，需要反复进行目标函数的评估，而近似模型是数学模型（如 Kriging 模型），评估一次近似模型需要的时间很短。无论采用什么方法，仿真函数还是需要评估的，可是评估次数要尽量少。当近似模型精度足够高时，就可以找到仿真函数的局部极值了。然后在局部极值点处构造采样区域并采样，并构建局部近似模型，最后进行局部搜索。反复进行此过程直到找到全局最优值为止。

本书的意义在于以下几个方面。

（1）拓展优化库的应用范围，提供给多物理建模、仿真及优化更加完善的工具，并且使得该平台更加实用化；对于动态响应优化问题的序列二次规划求解，提供并行化处理的策略，使得求解动态响应优化问题更有效率。

（2）针对机械系统动态响应优化问题进行研究，能够为面向机械系统的仿真动态响应优化提供基础支撑技术。

目前，求解机械系统运动微分方程或方程组，最常用的是有限差分法，而有限差分法在计算与时间相关响应的数值方法中占主导地位。在这种方法中，以初始条件开始用小时间步长计算与时间相关的微分方程，直到达到收敛要求为止。时间步的大小决定了计算的稳定性与准确性，这无疑限制了计算效率。而采用谱

元法，其将有限元法的处理边界和结构的灵活性与谱方法的快速收敛性结合了起来。在相同精度的情况下，谱元法能够采用较少的单元来减小计算开销。在一个单元内，将时间离散为与 GLL 多项式零点相对应的网格点，状态变量在这些点上进行 Lagrange 插值。从理论上分析，在一些点上插值，当这些点是正交多项式的零点时，获得的插值精度最高，而且在不增加单元数的前提下，可以达到谱收敛精度。

（3）面临目前极具挑战的计算昂贵的仿真动态响应优化问题，提供可收敛的近似优化解决方案。

由于仿真软件的成功应用，绝大部分真实物理实验都被仿真软件技术取代，它不仅快速而且廉价，这是所有工程技术领域期待已久的；近似优化技术的实现满足了工程师们最关心的最优设计要求，仿真软件不仅可以非常真实地反映物理特性，而且使得仿真时间不再是工程师们头痛的问题。面向仿真函数的响应面技术为仿真优化难题提供了有效合理的解决方案，基于响应面的仿真优化技术不但能保证获得设计空间的全局最优解，同时可以大大减少目标和约束仿真函数的估值（即仿真求解）次数。

（4）为计算昂贵的仿真动态响应优化提供一个从数据模糊聚类角度考虑的新思路。

模糊聚类是当涉及事物之间的模糊界限时，按一定要求对事物进行分类的数学方法。聚类分析是数理统计中的一种多元分析方法，它用数学方法定量地确定样本的亲疏关系，从而客观地划分类型。事物之间的界限，有些是确切的，有些则是模糊的。当聚类涉及事物之间的模糊界限时，需运用模糊聚类分析方法。模糊 c 均值聚类方法的目标是使群集数据的差异性最小化。一旦确定聚类数 c，该方法就自动识别群集中心并且将所有数据分到合适的数据群集里。在本书中，首先构造一个响应面近似模型，用采样技术得到一些点集合，通过仿真近似模型获得一些计算昂贵的函数值；其次，将所有函数值从小到大排序，选择前面 10%的点，去掉其他点；最后，对前面 10%的点集合应用模糊 c 均值聚类方法识别可能的最小值点的设计子空间。这样，我们将模糊聚类技术和全局近似优化有机地结合起来，为全局优化技术开辟了一条新的思路，给进一步研究基于仿真模型的动态响应优化奠定了非常重要的基础。

1.2　国内外研究概况

优化方法与计算分析仿真的集成将极大地影响产品设计[5]。正因为集成，所以面临诸多挑战，这些挑战主要是因为问题的高维、计算昂贵的分析仿真、函数的未知特性、仿真与时间相关模型的精度，以及与时间相关的约束需要在所有时

间点上满足而产生的。这里所说的未知函数就是所谓的黑箱函数；与时间相关的问题就是动态响应优化问题。学术界将高维、昂贵和黑箱（High-dimensional，computationally Expensive，Black-box）问题称为 HEB 问题。

HEB 问题在科学和工程实践中广泛存在。例如，高速民用运输机（HSCT）的机翼设计[6]，包括 26 个设计变量、4 个目标函数和 4 个工程约束，每执行一次 FLOPS/ENGGEN 和 ALCCA 仿真，大约需要 5 分钟；如果按两个水平的全因子分析，需要 67 108 864 次分析，在 IBM RISC6000 7012 model 320 工作站上计算需要整整 600 年。又如在汽车工业中，每执行一次碰撞分析平均需要 98 小时[7]，假设有 10 个变量，用两个水平的全因子设计，需要分析 1024 次分析，整整 12 年。

现代分析模型大部分是用商业软件构建的，如有限元分析和计算流体力学等。这些模型除了计算昂贵外，对设计者来说是隐含的和未知的。正是这隐含的函数给设计优化造成了极大的困难[8]。当设计变量数增加时，计算要求将按指数级增加[9、10]。由问题维数引起的困难称为“维数灾难（curse-of-dimensionality）”。

1.2.1 动态响应优化

机械结构大都在随时间变化的动载荷作用下工作，机器的各种动态性能均表现为时间的函数。为了使机器的动态性能指标达到极值，有必要进行动态响应优化，这样一方面需要处理在整个时间区间上的载荷变化，精确求出整个时间区间上的响应，满足与时间相关的约束，而且不漏掉任何一个响应最值；另一方面，需要采用小时间步计算时间响应，防止约束在任意两个时间点之间失效。优化器每迭代一步，目标函数、约束和灵敏度需要重新计算，因此在优化计算时，响应求解必须是高效率的。

有限差分法在计算与时间相关响应的数值方法中占主导地位。在这种方法中，从初始条件开始用小时间步长计算与时间相关的二阶微分方程，直到达到收敛要求为止，时间步的大小决定了计算的稳定性与准确性，这无疑限制了计算效率；而多个瞬态周期以后，有限差分法会达到稳态循环，此时计算周期响应，计算效率会相对比较高。但是当受脉冲激励时，基于频率的方法解决响应快速变化的精度不高。在文献[17]中，对于脉冲激励的动力系统，应用谱元法在半连续附近采用了 h 型精细方案，并且将微分方程或方程组转化为代数方程组，在不增加单元数的前提下，达到了谱收敛精度。

在文献[62]中，提出了将设计变量和位移响应、设计变量和位移响应与速度响应、设计变量和位移响应与速度、加速度响应分别作为优化变量的三种方案，应用有限差分法近似与时间相关的约束，运动微分方程被作为等式约束处理，这

样能用显式表达式给出优化变量的梯度，获得梯度信息的效率高。可是将一个少变量问题变成高维问题，不仅求解困难，而且很难收敛。对于动态响应最大值最小问题，由于在迭代过程中最大响应是振荡的，直接处理目标函数很难收敛[11]，所以 Dong-Hoon Choi 提出了直接处理目标函数的方法，解决了收敛的困难，该方法通过预先给定一个全局最大响应的系数α（$0<\alpha\leqslant 1$），在优化过程中同时使大于预先给定的全局最大值的局部最大值最小。

Chung-Ho Wang[12]提出了一个新的动态响应优化方法，即主要部件分析和灰色关联模型的多准则评估方法。这种方法考虑了获得无关部件的多质量特征之间的关系，且部件的优化因子水平由灰色关联模型决定。在文献[13]中，动态模型包括一阶常微分方程和代数方程，且将连续动态响应优化问题转化为大型非线性参数约束优化问题，并应用序列二次规划法求解。在文献[14]中，对自由飞行的空中交通运输中的碰撞问题，即动态响应优化问题，将其转化为有限维的非线性规划问题，然后应用内点法求解。

有限元法最大的特点是它将运动微分方程转化为代数方程[15]，这样就可以获得响应对设计变量的更多显式关系，有效地计算响应的灵敏度。1984 年，Patera 提出了谱元法[16]，它兼有有限元法处理边界和结构的灵活性与谱方法的快算收敛性的特性。在一个单元内，将时间离散为与 GLL（Gauss-Lobatto-Legendre）多项式零点相对应的网格点，在这些点上进行 Lagrange 插值，在整个时间区间上解微分方程，包括瞬态响应和稳态响应；当求解脉冲激励响应时，在整个时间区间上应用谱元法进行离散，通过在激励突变附近增加插值次数（p refinement），或者减少单元尺寸（h refinement）体现了其局部灵活性[17]。

对承受瞬态载荷的结构优化在 1970 年就开始了研究。2006 年，Kang 等[18]首次把动态响应优化作为优化的一个分支。在其优化中，设计变量每迭代一次，时间响应需要计算一次，同时约束必须在整个时间区间内被满足。处理时间约束有几种方法：一种是只在响应的全局最大值处满足约束；另一种更加稳定的方法是在更小时间步长上满足约束，使中间点处的约束失效不可能发生。在准静态方法中应用这种方法使得多个准静态载荷满足约束而不是动态载荷[19]。在这些方法中，约束数大大增加。因为在优化迭代过程中，要计算这些约束的灵敏度，所以优化耗费也在增加。一种更加有效的方法是只在响应的局部极值点处处理约束，这样减少了约束数量和计算约束对设计变量的灵敏度的次数[20]。

1.2.2　基于仿真的优化研究概况

在仿真优化迭代过程中，需要调用仿真程序来计算目标函数和约束函数的值。响应面方法是提高仿真优化效率的有效途径。响应面（Response Surface）是指输

出响应变量 Y 与一组输入变量 X (x_1, x_2, …, x_n)之间的函数关系。通常，响应面反映的是某个计算密集的复杂源模型（如多领域仿真模型、FEA 模型、CFD 模型等）的近似模型。因此，响应面又称为代理模型（Surrogate）或元模型（Meta-model），即模型的模型。

响应面方法（Response Surface Method，RSM）是指构造源模型的响应面，以解决源模型的设计或分析等问题的近似方法。据报道[7]，福特汽车公司进行一次汽车碰撞模型仿真分析需要的时间大约为 36～160 小时，要实现该模型的两个变量的设计优化，假设平均需要 50 次迭代寻优，而每次迭代需要进行一次仿真计算，这样获得该优化问题的解大约需要 75 天到 11 个月。同样，要实现 FEA 模型或 CFD 模型的设计优化可能需要更长的时间，这在实践中几乎是不可接受的。因此，在过去的 20 年中，响应面方法应运而生，而且得到了快速发展。该方法能够减少优化迭代过程中原模型的仿真次数，而且响应面都是基于采样点数据构造的，而采样点估值计算都是彼此独立的（传统的优化迭代过程是序列估值的），可以方便地通过并行计算来获得，因此，该方法可以大大提高复杂分析模型的设计优化效率。

根据文献[21]，RSM 的作用包括以下 4 个方面。

（1）模型近似：这是 RSM 的基本功能。建立一个复杂源模型在其全局定义域内的响应面近似模型，可以利用该近似模型实现新的未知设计点的快速估值。

（2）设计空间探索：建立的响应面模型可以帮助工程师或设计人员进行参数实验、灵敏度分析，以及将响应变量与输入参数之间的函数关系可视化，从而帮助工程师更好地理解源模型的特性。

（3）优化问题的准确表达：基于对响应面模型的设计空间探索，特别是灵敏度分析，可以帮助设计人员构造更加准确的设计优化问题。例如，可以从设计变量集合中剔除那些非敏感的参数，从而减少设计变量的维数。根据参数试验也可以缩减搜索区间，从而减小采样区间，进而减少优化迭代次数。同样的，通过分析，一个多目标设计优化的问题可能被简化成单目标优化问题，而原以为是一个简单的单目标设计优化问题通过设计空间探索，发现可能需要建立多目标设计优化问题才能得到解决。

（4）优化方法的支持：这是目前 RSM 的主要应用领域。利用建立的响应面模型可以辅助完成各种涉及源模型仿真的设计优化问题，如全局优化、多领域仿真优化、多目标优化、多学科设计优化，以及概率设计优化（包括可靠性优化、稳健优化等）。

响应面方法的一个重要环节是构造响应面模型。响应面模型的种类较多，分别适合不同的需求。常用的有多项式回归模型（Polynomial Regression Surrogate，PRS）、Kriging 插值模型、径向基函数（Radial Basis Functions，RBF）模型、支

持向量回归（Support Vector Regression，SVR）模型、神经网络（Neural Network，NN）模型，以及基于 RBF 和 NN 混合的 RBNN 模型；还有基于样条的多元自适应回归样条（Multivariate Adaptive Regression Spline，MARS）模型、BMARS（B 样条 MARS）及 NURBs 模型，其他的还有归纳学习（Inductive Learning）模型、最小插值多项式（Least Interpolating Polynomial，LIP）模型等。

要构造一个响应面模型，第一步是在设计空间内采点，形成设计点集 S；再进行仿真计算（也称“昂贵”计算，Expensive Calculation）获得响应数据集 Y；最后是根据不同的算法由 S 和 Y 构造不同的响应面模型。计算机试验设计方法提供了各种采点的策略，主要包括两大类：边缘分布型和全空间分布型。边缘分布型也称经典采样方法，利用该方法采样，其采样点主要分布在设计域的边界处附近，典型的边缘分布型采样方法有全因子/部分因子试验（Full/Fractional Factorial Design）、中心复合试验（Center Composite Design，CCD）、Box-Behnken 等，还有 Taguchi、D-Optimal、Plackett-Burman 方法等。全空间分布型（Space Filling）是指采样点布满整个设计域，该类型的采样方法有简单网格、拉丁超立方设计（Latin Hypercube Design）、正交表（Orthogonal Array），以及随机采样、一致设计（Uniform Design）、混杂网络（Scrambled nets）、蒙特卡罗仿真和 Hammersley 序列设计等。

一般的，一次采样构造一个响应面模型是不合适的，原因在于：如果采样点过多，则构造过程费时而且可能影响使用；如果采样点过少，则构造的响应面不够准确，难以满足应用需求；另外，由于源模型的性态未知，难以确定合适的采样方法。解决此问题的方法是基于序列自适应采样以构造序列响应面。序列自适应采样的主要思想是根据近似值与真实值之间的误差大小来确定采样点的疏密，序列探索试验设计（Sequential Exploratory Experiment Design，SEED）[22]方法是此类试验设计方法的代表，而 i-Sight 软件中使用了模拟退火算法来进行自适应采样。

当使用响应面近似方法解决实际工程问题时，应综合考虑如下 5 个方面的因素。

（1）响应面的精确度。毫无疑问，响应面模型的精确度是近似的基本要求。

（2）源模型的仿真估值次数，即构造响应面所需的总的采样点数目。由于每次估算的计算费用较高，所以需要限制源模型的仿真次数。在相同精度情况下，源模型仿真估值次数越少，该响应面模型的构造效率越高。

（3）构建和优化响应面的时间。如前所述，响应面的构造往往是一个逐步精细的过程，自适应序列采样构造逐步精细的响应面模型是目前响应面方法的一个研究热点，特别是在采样点逐步增多的情况下，如何快速更新响应面以实现响应面的增量构造算法是各种响应面方法值得探索的课题。

（4）响应面占用的存储空间。显然，响应面模型本身所占的内存空间越大，则构造过程越慢，利用其进行估值也越慢。因此，在同等情况下，响应面模型本身所含的信息越少（也即所占的内存空间越小）越好。

（5）利用响应面模型对给定点的估值速度来建立响应面的最终目的是使用其进行估值，而且通常这个估值过程被称为“便宜计算”（Cheap calculation），因此它在应用于优化迭代过程中被大量地执行。由此可见，如果对给定点的估值速度太慢，就会使得理想的“便宜计算”变得不“便宜”了。

对于一个非线性很强、设计空间又很大（高维）的源模型或源函数而言，使用一个完整的响应面来近似它往往很困难。局部响应面是响应面技术的研究方向之一，即从设计空间中找出全局最优点所在的子区域，和全局响应面相比，在此子区域内建立响应面有明显的精度优势。因此，缩减搜索区域成为目前优化领域研究的热点。

处理高维问题的策略包括提高计算机的配置、减小设计空间、筛选重要设计变量、分解设计问题、映射和可视化设计变量或设计空间，它们从不同的侧面处理了高维问题面临的困难。

分解是指重新表达原始问题，并将其表达为独立的或协调的小规模的子问题。分解法已经取得了很好的成果并广泛应用于解决复杂工程问题[23~25]。在工程中，分解可以分为产品分解、过程分解和问题分解。产品分解是指将一个产品分为多个物理模块，其缺点是描述各物理模块边界是主观的；过程分解应用于包括元素和信息流的问题；问题分解是指将复杂问题划分为不同的子问题。在解决设计问题时，基于分解的设计优化促进了非线性优化技术的应用。

一般采用矩阵描述问题之间的关系，称之为关系矩阵。因此，通过分解关系矩阵，问题就被分解了。通常，有两个基本关系矩阵：设计结构矩阵（Design Structure Matrix，DSM）和函数独立矩阵（Function Dependent Matrix，FDM）。DSM 是方阵，行和列相等，表示一套单一的对象[26]。FDM 的行和列不同，分别表示两组对象的关系。经常采用数学工具分解矩阵，如图分解、聚类分析和优化。通常，这些算法取决于如何建立分解模型。有三种主要的分解模型类型：第一种类型是超图[27]、网络可靠性[28]和整型规划[29]；第二种类型是智能方法[30]；第三种类型是聚类方法[31]。对于 DSM 而言，聚类方法用以揭示结构关系；排序算法用以揭示信息关系。对于 FDM 而言，聚类有益于设计优化和分类技术。

分解模式有两类：理想分解和协调分解。理想分解将关系矩阵对角化，使之成为块之间没有任何交叉的完全独立的模块。如果设计完全遵循公理设计理论[32]，可以获得理想分解。协调分解是更切合实际的分解模式，包含块与块之间存在的交叉关系。

在实际的工程设计中，设计工程师处理优化问题的初始值，一般采取很保守的方法，即将设计变量的下限尽量取得小一些，上限尽量取得大一些，这样可以保证不会漏掉可能的最优值点，就像渔夫尽量撒大网一样，能钓到尽可能多的鱼。然而对于局部优化算法而言，设计变量取不同的初始值和不同的取值范围可能有不同的结果，当然对于全局优化算法来说，不存在这个问题，但是当取值范围变大时，目标函数和约束函数的仿真次数会增大，因为绝大多数工程问题仿真一次非常耗时（耗时有时是无法容忍的甚至是不可行的），这样会增大仿真次数。对于工程师的这种做法，经常是由于在定义优化问题时，缺乏目标函数与约束函数之间的交叉，以及缺乏目标函数与约束函数的性质形成的。许多学者提出了区域缩减的优化设计方法。常用的区域缩减法有变量筛选法（Variable Screening，VS）、阈值法（Threshold Method，TM）、重要区域识别法（Identification of Important Region，IIR）、粗糙集法（Rough Set，RS）、移动指标策略法（Move Limit Indicator Strategy，MLIS）、信任域法（Trust Region Method，TRM），以及近似单峰值区域消除法（Approximated Unimodel Region Elimination，AUMRE）和域优化算法（Domain Optimization Algorithm，DOA）。

变量筛选[33]是一种重要的区域缩减方法，一旦维度减小，其变量的空间将按指数级缩减。灵敏度分析或参数试验是变量筛选的可靠方法，即在一定的区域内，观察响应变量 y 与某个参数 x_i 之间的关系，如果关系曲线平坦或在很小的区间内波动，则可以认为该参数 x_i 为非重要参数，可以被筛选掉。基于某个响应面模型的“便宜”计算可方便地实现灵敏度分析，在局部区域内执行灵敏度分析则可以判别某参数在局部区域的重要性。基于采样点数据也可以判别变量的重要性。变量筛选可以通过 ANOVA 分析实现[34]。MARS 模型在其构造过程中可以自动进行变量的筛选[35]。

阈值法（Threshold Method）[36]是由 Wang 和 Simpson 于 2004 年提出来的，其基本思想是：首先构造一个响应面近似模型，用采样技术得到一些点集，通过仿真近似模型获得每一点的函数值，然后选择一个阈值，当函数值大于阈值时，则去掉该点，最后对剩余的点集合应用模糊 c 均值聚类方法识别可能的最小值点的设计子空间。

确定性优化方法[37]通过穷举搜索整个设计空间 D 或有限的密集子集 D 来找到全局最优值点。很明显，为了能保证在有限迭代步骤内找到最优值点，不得不对目标函数 f 施加某些限制，对于全局最小值点在任意小的子空间内的函数来说，很容易构造目标函数。最著名的假设是 Lipschitz 假设，即给定一个 Lipschitz 常数，对于所有 $x,y\in S$ 都满足 $|f(x)-f(y)|\leqslant\|x-y\|$，然而 Lipschitz 假设在实践中很少得到验证。因此，有学者们提出随机法，即通过给优化器提供一个有一定概率的有希望包括最优值点的区间来实现全局最优。

传统的优化过程从设计点开始，然后计算分析目标函数和约束函数，直到最优值找到为止。因此，这是从设计空间到值空间的搜索，称为向前搜索方法。相对应的从值空间到设计空间的搜索，称为向后搜索方法，即设计空间缩减[38]。

最近，在多学科设计优化领域发展了一种逐渐减小设计空间的方法。设计空间是由设计变量的边界合并而成的一个 n 维超立方体。这个超立方体最初是由设计人员给出的，也就是优化算法的搜索范围，其对优化耗时和是否能找到最优点有很大影响。当然，可以减少设计变量的维数，但是这往往比较困难[39]；也可以假设维数不能减少，则必须减小设计空间。

粗糙集理论是处理数据的分类分析，并且是一种严格的数学工具，因此粗糙集理论非常适合处理优化中的空间减小问题，它是一种从值域到设计域的不依赖于设计空间的设计方法。采用拉丁超立方（Latin Hypercube Sample，LHS）等采样技术在设计空间采样，得到 n 个 (x_1,x_2) 数据点，再仿真得到 $f(x)$的对应值。对函数值进行变换后，应用粗糙集理论的决策值进行操作，获得重要空间，这个重要空间可能包括一个或多个，连续或不连续的子重要空间，然后利用空间重叠系数来判断是否收敛。

Vassili Toropov 等于 1996 年提出了移动指标策略[40]（Move Limit Indicator Strategy），其基本思想是：在每一次搜索之前，首先要确定搜索区间的大小和位置。该策略是通过近似的功能质量指标、局部最优点在当前搜索子空间的位置、有关移动历史、收敛准则和有关最活跃的约束等 7 个指标来控制的，它们能很好地体现近似的质量和优化迭代收敛的过程。

最早的信任域法（Trust Region Method）是由 Levenberg 于 1944 年提出来的，其基本思想是：在局部可信赖的区域内，先构造近似模型，再解局部优化问题，解得的结果有两种情况，一种是可信，即 $f(x_{k+1})<f(x_k)$，另一种是不可信，即 $f(x_{k+1})>f(x_k)$；当可信时，增大信任域半径，再构造局部近似模型，解局部优化问题；当不可信时，减小信任域半径，再构造局部近似模型，解局部优化问题；重复上述过程直至收敛。

信任域法的概念和解释在不断地发展，其思想逐渐地推广到各类应用中。它在 60 多年的研究中不断成熟起来。例如，在应用最广的序列二次规划法（Sequential Quadratic Programming，SQP）中，成功地应用了信任域法的思想。由于仿真软件的成功应用，绝大部分真实物理实验都被仿真软件技术取代，不仅实现了快速而且廉价，这是所有工程技术领域期待已久的；然而工程师们最关心的是得到最优的设计。虽然仿真软件可以非常真实地反映物理特性，但是仿真时间是最令他们头痛的，为了解决这个问题，仿真优化出现了，其中优化算法显得极其重要。在仿真优化中，如果直接采用现有算法，如基于梯度的序列二次规划算法、启发式

优化算法遗传算法（GA）等，仿真次数非常大，常常是无法容忍或是不可能的。Schmit 在 1974 年提出了基于近似响应面的优化。其在应用中体现出了很大的优越性。而信任域法在管理优化中的响应面方面体现出了优越性。Conn，A. R.在 1988 年、1991 年深入研究了无约束优化中的信任域法，更重要的是证明了其严格的最小值。J. F. Rodriguez 在 1998 年将其扩展到约束优化[41、42]，同时应用试验设计（Design Of Experiment，DOE）进行了采样。

近似单峰值区域消除法（Approximated Unimodal Region Elimination，AUMRE）[43]由 Younis，A. A.在 2008 年提出，是基于试验设计、区域缩减和响应面模型的一种全局优化方法。该方法的使用步骤包括：用实验设计数据将感兴趣的区域划分为几个单峰值区域；识别和排序有希望的区域，即有很大概率包括全局最优点；在这个区域，用一些附加的实验数据构造响应面；搜索最优点，删除处理过的区域，然后移到下一个有希望的区域。通过避免重复搜索，该方法减少了目标函数评估次数和优化计算工作量，获得了全局最优。

域优化算法（Domain Optimization Algorithm，DOA）[44]是由 Vinícius V. de Melo 在 2007 年提出来的，属于一种搜索空间缩减算法（Search-space Reduction Algorithm，SRAs）。在调用一个优化器之前，一个搜索空间缩减算法给定一个或多个有希望的区域，而不是在优化过程中识别有希望的区域。域优化算法（DOA）通过识别出最大可能找到全局最优值点的区域提供给全局最优算法，作为全局最优算法的初始搜索空间，帮助找到全局最优值点。此算法能够减小初始搜索空间，排除不利的区间。该算法用很高的概率保证包括全局最优值点的区域没有被删除，而不是构造复杂的概率模型来确定重要区域。文献[44]中采用遗传算法（Genetic Algorithm，GA）和粒子群算法（Particle Swarm Optimization，PSO）算法来执行经 DOA 处理的优化问题，当然也可以采用其他优化算法，但是无论采用哪种优化算法，DOA 的准确性和效率均会直接影响优化结果。

1.3　本书研究内容、创新点与组织结构

1.3.1　本书的主要研究内容

本书针对基于仿真黑箱函数模型的动态响应优化的工程实际问题，研究了有效且高效的优化技术。本书的主要研究内容如下。

（1）研究基于仿真的动态响应优化中 SQP 算法的并行处理与调度策略，对算法在理论上的可行性做了深入研究；采用机群系统构建了并行仿真优化环境，在自主研发的多领域统一建模与仿真平台优化库上开发了并行优化模块。

（2）研究基于时间谱元法的机械系统动态响应优化设计；深入探讨在时间域内的离散动态响应，将运动微分方程组转化成代数方程组，精确解出瞬态响应；引入人工设计变量，详细分析了两种处理约束方法的优缺点。

（3）研究多变量自适应回归样条构造全局响应面，采用移动极限策略求解基于仿真的动态响应优化问题。多变量自适应回归样条（MARS）具有很强的自适应性，且对模型预测的精度也比较高，因此在工程实际中得到了广泛的应用。移动指标策略法（MLS）在求解近似优化子问题后，必须确定一个新的搜索子区域，即必须指定它的大小和位置；它主要包括 7 个指标，它们反映了近似的质量和优化收敛历史。

研究数据驱动在计算昂贵的仿真动态响应优化中的应用。数据驱动主要是解决随机优化中的采样重复。当采样和数据库中的点重复或很近时，就需要存取数据库而不是评估计算昂贵的仿真模块，这样就可以节省很多时间。对于近似模型技术，数据驱动起着重要作用。

将信任域法与多变量自适应回归样条和数据驱动技术结合起来，可解决计算昂贵的仿真动态响应优化问题。

（4）研究基于模糊聚类和近似模型的全局动态响应优化方法。对于计算昂贵的仿真动态响应优化问题，研究的重点在于减少仿真函数的评估和计算昂贵分析的次数。因为在优化过程中，需要反复进行目标函数的评估，而近似模型是数学模型，评估一次近似模型需要的时间很短，所以可应用 Kriging 模型作为仿真模型的近似模型。当然，仿真函数必须要评估，但是评估次数一定要非常少。当近似模型非常接近仿真函数时，就可以获得近似模型的局部峰值，然后在这个局部区域采样，评估仿真函数，构造局部近似模型，最后进行局部搜索。反复进行这个过程直到找到全局最小值为止。

1.3.2 主要创新点

本书对基于仿真的动态响应优化和近似优化方法进行了研究，主要创新点有以下几点。

（1）在 CAD 中心优化库上实现了基于仿真模型的动态响应序列二次规划算法的并行化，并抽象出一类等式约束离散变量优化模型，拓展了优化库的应用范围，而且使得优化库的实用性更强了。注意，并行计算中，细分计算任务和充分利用现有计算资源是问题的关键；通过工作池，应用集中式动态负载平衡技术可实现并行计算梯度；将多处理机任务转化为等式约束离散变量优化问题，应用方向差商法求解。

（2）将时间谱元法应用到机械系统动态响应优化中。应用谱元法求解机械系

统动态响应，改善了传统求解动态响应时误差大的缺点，达到了谱收敛精度。因此，动态响应优化可以在超曲线或超曲面上找到满足所有时间约束的变化的目标函数。在处理约束上，本书采用了 GLL 点法和关键点法两种方法，并比较了这两种方法的优缺点。

（3）提出多变量自适应回归样条（MARS）与移动极限策略（MLS）结合起来研究解决 HEB 问题的动态响应优化算法。本书利用 MARS 构造全局响应面，应用 7 个指标值来确定下一次迭代子区间的位置和大小，并将该子区间作为本次迭代的设计域，应用现有优化器求解该子问题，这样反复迭代直到收敛。

（4）将多变量自适应回归样条与信任域法和数据驱动技术结合起来研究解决 HEB 问题的动态响应优化算法。关键是数据驱动与序列响应面协调。是否进行昂贵仿真评估采取的策略：只有当最优点在数据库中或有很近的点在数据库中，才用数据库中的点代替当前最优点，而不需要再进行计算昂贵的仿真分析，如果不存在这样的点，只能通过计算昂贵的仿真分析来获得最优点的目标函数和约束函数的值，这样极大地提高了仿真优化的效率。随着迭代次数的增加，数据库不断地丰富起来，对仿真优化更加有利。

（5）将数据聚类应用于全局动态响应优化。首先通过 Kriging 近似模型获得模糊聚类的对象，再由模糊聚类中心与其几何中心确定重要区域的中心，前两次迭代结果确定重要区域的半径。对于昂贵计算和黑箱函数优化问题，研究的重点在于减少黑箱函数的评估和计算昂贵分析的次数。在优化过程中，需要反复进行目标函数的评估，而近似模型是数学模型，评估一次近似模型需要的时间很少。当然，黑箱函数必须要评估，但是评估次数一定要非常少。当近似模型非常接近黑箱函数时，就可以获得近似模型的全局最优点，即为动态优化问题的全局最优解。

1.3.3　组织结构

全书共分为 6 章，具体章节的内容安排如下：

第 1 章，绪论，包括课题目的和意义，文献综述等；

第 2 章，动态响应优化问题的并行化处理；

第 3 章，基于时间谱元法的动态响应优化算法；

第 4 章，多元模型自适应与时间谱元法结合的动态响应优化；

第 5 章，基于 MARS 的动态响应优化算法；

第 6 章，基于模糊聚类的全局动态响应优化算法。

各章节的组织结构如图 1.6 所示。

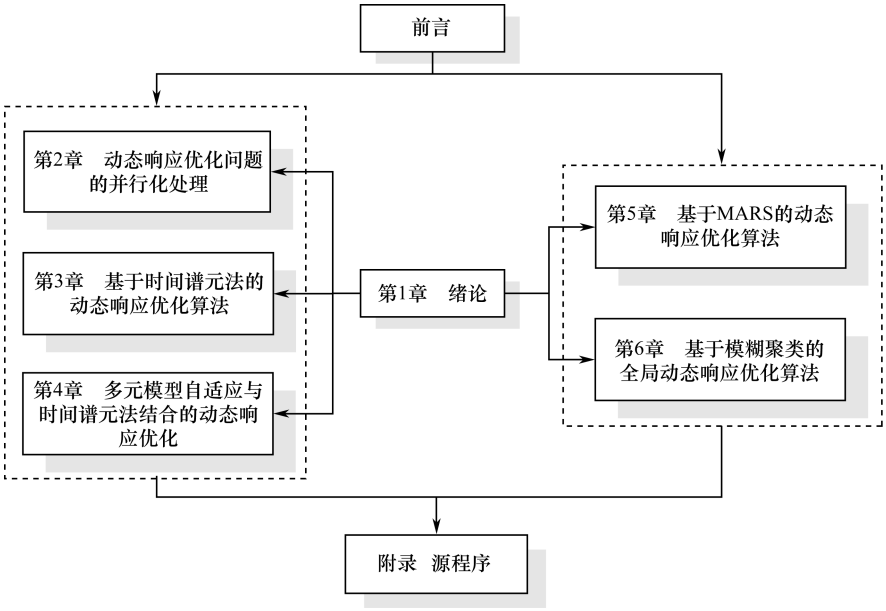


图 1.6　本书各章的组织结构

第 2 章　动态响应优化问题的并行化处理

目前比较成熟的优化算法[45~47]有约束变尺度法、拟牛顿法、Hook-Jeeves 算法和 GA 算法等。约束变尺度法又被称为序列二次规划（Sequential Quadratic Programming，SQP）算法，具有收敛快、效率高、可靠性与整体收敛性好、适应能力强等一系列优点[48]，并被成功地应用于工程优化问题的求解。序列二次规划算法通过求解一系列的二次规划子问题（Sequential Programming，QP）来获得原问题的最优解，而其二次规划子问题的精度依赖于原问题的目标函数和约束函数梯度值的精度，并对问题的收敛性有较大影响。对于基于仿真模型的动态响应优化而言，目标函数和约束函数没有解析表达式，因此对其梯度计算只能采用差分法。当采用差分法计算梯度时，由于每次迭代需要多次进行仿真函数的评估（求值次数等于设计变量维数的两倍乘以实例数+1），而且每次评估仿真模型都需要计算整个时间域上的模型值，这样优化求解过程非常耗时，所以研究 SQP 并行优化算法很有必要。

2.1　动态响应优化算法的并行化

2.1.1　SQP 算法的并行化

对于 SQP 算法的并行化，国内外有关研究已取得了一定的成果。High[49,50]从全局优化、函数评估、梯度评估与 Hessian 矩阵的有限差分计算等方面讨论了 SQP 算法的并行化，并对许多算例进行了计算。张帆[51]讨论了大规模化工过程系统中，优化算法 SQP 的并行化和机群计算效率的问题，并通过算例证明了其合理性。陈忠[52]叙述了关于非线性规划问题的求解有几种主要并行思想，简要地综述了数值优化的进展。总之，SQP 算法的并行化研究主要集中于在传统优化的基础上结合具体的问题，而基于多学科仿真多实例、多目标的 SQP 优化算法的并行化研究甚少，且绝大多数的研究是针对专用并行机的。因此，本章就从基于多学科仿真的多实例、多目标一般性优化出发，研究 SQP 梯度计算并行化和并行仿真评估。

在优化算法中，梯度计算是最耗时间的步骤，同时也是设计优化的瓶颈[53]，SQP 算法也不例外。然而，因为各个变量的灵敏度计算是互相独立的，所以它很适合并行化。

SQP 并行优化算法的步骤如下。

（1）给定初始值 X^0、λ^0，其中 X 是设计变量，λ是 Lagrange 乘子。

（2）并行仿真评估、并行计算梯度值[50]：

① 并行仿真评估；

② 并行计算梯度值。

在这里应用集中式动态负载平衡技术[54]实现了并行计算梯度值。工作池技术能很好地实现动态负载平衡，如图 2.1 所示。

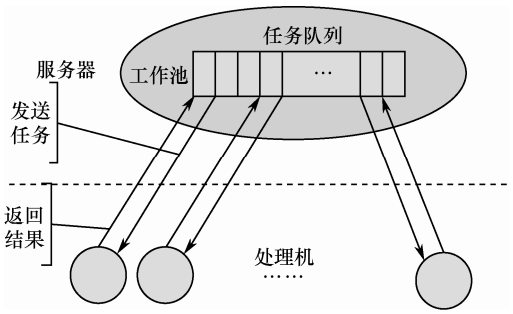


图 2.1　集中式工作池

（3）构造二次规划子问题。

（4）求解二次规划子问题，确定新的乘子向量λ^k和搜索方向 d^k。

（5）沿 d^k 方向进行一维搜索，确定步长α_k，得到新的约束极小值点。

（6）满足收敛精度[50]：

$$\| \nabla_x L[X^{(k+1)}, \lambda^{(k+1)}] \|_2 \leqslant \varepsilon_1 \tag{2.1}$$

式中，L 是 Lagrange 函数；ε_1 是收敛精度。

或满足下式：

$$\begin{cases} c_i[X^{(k+1)}] \leqslant \varepsilon_2 , i \in E \cup I \\ \dfrac{\left| f[X^{(k+1)}] - f[X^{(k)}] \right|}{f[X^{(k)}]} \leqslant \varepsilon_3 \end{cases} \tag{2.2}$$

式中，f 是优化的目标函数；c 表示优化约束；E 表示等式约束集合；I 表示不等式约束集合；ε_2,ε_3 是收敛精度。

则停止计算，否则转到步骤（2）。

（7）求解 Hessian 近似阵[50]，令 k=k+1，返回步骤（3）。

2.1.2　并行调度算法

1. 问题描述

在分布式环境下，因为每一个处理机的性能都不相同，所以通常需要用多个

处理机来完成一个任务，我们把这种任务称为多处理机任务。通常，k 个处理机的多处理机系统表示为 $P=\{p_1,p_2,\cdots,p_k\}$，多处理机任务集合表示为 $J=\{j_1,j_2,\cdots,j_n\}$，其中 $j_i(1\leqslant i\leqslant n)$ 是二元属性 (s_i,t_i)，$s_i\subseteq P$ 为执行任务 j_i 要求具有的一组处理机，一般称为处理机模式[55]，t_i 为该组处理机一起处理这任务所花费的时间，称为时间长度。

并行调度就是为提交给系统的每一个任务分配一定的资源，并指派占用这些资源的起止时间，目的是在总体上达到时间跨度最小，这是典型的多处理机任务调度问题，记为 $S_k\left|\text{fix}\right|C_{\max}$ [56]，S_k 是 k 个独立的处理机，fix 是调度实例的约束，包括处理机的分配方式、任务之间的优先性、资源需求等方面的约束，$C_{\max}$ 表示要使最后完成时间达到最小，提高系统的总吞吐量。

在多实例、多目标并行优化中，调度模型 $S_k\,|\,\text{fix}\,|\,C_{\max}$ 有些特殊，其中 S_k 和 $C_{\max}$ 的意义不变，而 fix 表示每一个任务只需要一个处理机。例如，3 个变量、3 个实例的多实例、多目标优化问题，每迭代一次有 21 个并行任务，可以描述为 $J=\{j_1,j_2,\cdots,j_{21}\}$，其中，$J$ 中的每一个任务都只需要一个处理机。

现在研究 N 个处理机环境下的并行优化问题。假设优化问题有 n 个设计变量（调节参数），m 个模型实例，则多实例、多目标并行优化并行任务可以描述为

$$J_{\text{iter}}=\{j_i=(s_k,t_i)\,|\,i=1,2,\cdots,m(2n+1),s_k=p_i,1\leqslant k\leqslant N\}\tag{2.3}$$

式中，iter 是优化迭代次数。

由于优化的每一次迭代任务的模式都相同，只是其中的数值不同，所以只分析其中的一次迭代就够了。执行时间是 t_{iter}，且

$$t_{\text{iter}}=\max\{\sum_{k=1}^{q_i}t_k,i=1,2,\cdots,N,q_i<m(2n+1)\}\tag{2.4}$$

显然，任意一个调度的时间跨度不小于 N 个性能最好处理机执行的总时间，由此得到如下一个下界限：

令 $B=\min\{t_k\,|\,k=1,2,\cdots,N\}$，则 $\text{Opt}(J)\geqslant B$，$\text{Opt}(J)$ 表示优化函数。

2. 调度算法

多实例、多目标并行优化 $S_5\left|\text{fix}\right|C_{\max}$ 问题的一个调度算法如下。

输入：任务实例 $J=\{j_1,j_2,\cdots,j_{21}\}$，其中 $j_i=(p_i,t_k),i=1,2,\cdots,m(n+1),k=1,2,\cdots,N$。

输出：实例 J 的一个调度。

调度算法步骤如下。

（1）得到所有连接的客户端，评估其性能，从高到低排序，生成处理机性能队列。

（2）以单位性能任务数和处理机性能数为依据分配任务。

将单位性能处理机看作有两个线程的计算节点，则其他处理机以其性能数为依据，决定其启动的线程数。例如，5 处理机系统，其性能数分别为 1、1.5、2、1、2.5，则它们同时启动的线程数分别为 2、3、4、2、5。

每一个节点处理机可以同时启动两个或两个以上的线程，实现二级并行计算。由于任务数大，要分散操作多次才能完成所有任务。每一次分散操作的总任务数为所有处理机可以同时启动的线程数之和。例如，5 处理机系统，可以同时启动的线程数为 16，则一次分散操作只能完成 16 个任务。这样任务分配可以描述为

$$J=\{\{j_1,j_2,\cdots,j_L\}_1,\{j_{L+1},j_{L+2},\cdots,j_{2L}\}_2,\cdots,\{j_x,j_{x+L},\cdots,j_{m(2n+1)}\}_i,m(2n+1)>N\} \quad (2.5)$$

式中，m 为模型实例数；n 为设计变量数；N 为独立的处理机个数；L 为所有处理机可同时启动的线程数；i 为完成 $m(2n+1)$ 个任务需要分散操作的次数。

（3）分散操作。

分散任务并执行，即将任务队列中的每一组任务按其线程数分别发送给调度算法步骤（1）中建立的处理机队列对应的处理机。最后一次分散操作比较特殊，可能所有处理机的总线程数大于所剩任务数，如果此时按性能好的处理机优先进行分散操作，如 5 处理机系统，其性能数分别为 1、1.5、2、1、2.5，则它们同时启动的线程数分别为 2、3、4、2、5，问题规模为 21 个任务，应该进行两次分散操作，第一次每一个处理机分配和同时启动线程数相等的任务数，共分配了 16 个任务，第二次 5 个处理机的 16 个线程只有 5 个任务，这时就有两种策略：①每一个处理机分配一个任务；②只分配给性能好的部分处理机。显然这两种策略所花费的总时间是不相等的，下面对此问题进行抽象描述。

此问题的一般性描述为：假设有 k 个处理机，按其性能从高到低排序为 $p_1,p_2,\cdots,p_k$，它们分别可以同时启动的线程数为 $L_1,L_2,\cdots,L_k$，分别启动一个线程计算所花时间为 $t_1,t_2,\cdots,t_k$，任务数为 J，当 $J<\sum_{i=1}^{k}L_i$ 时，问题就转化为

$$\begin{cases}\min\max\{t_ix_i\}\\ \text{s.t.}\ \sum_{i=1}^{k}x_i=J,i=1,2,\cdots,k\\ x_i\in S_i;S_i\in S\end{cases} \quad (2.6)$$

式中，x_i 表示第 i 个处理机启动的线程数；常数 $J<\sum_{i=1}^{k}L_i$；t_i 都为给定的常数，$i=1,2,\cdots,k$；x_i 只能取整数。

假设 $f(x)=\max(t_ix_i)$，$G(X)=\sum_{i=1}^{k}x_i$，$i=1,2,\cdots,k$，则模型变为

$$\begin{cases} \min f(X) \\ \text{s.t.} \quad G(X)=J, i=1,2,\cdots,k \\ X \in S_i; S_i \in S \end{cases} \tag{2.7}$$

这是等式约束离散变量优化问题。应用方向差商法[57]可求得最优解，即最优分配策略。

例如，有 8 台处理机，它们可同时启动的线程分别为 2、3、4、2、5、6、3、5，任务是 3 个设计变量、3 个模型实例的多实例、多目标优化问题。每一次迭代的任务数为 21，8 台计算机可以同时启动的线程总数为 30，假设每个处理机启动一个线程计算一个任务需要的时间分别是 5，4，3，5，2，1，4，2s，则数学模型为

$$\begin{cases} \min \max\{5x_1, 4x_2, 3x_3, 5x_4, 2x_5, x_6, 4x_7, 2x_8\} \\ \text{s.t.} \quad \sum_{i=1}^{8} x_i = 21 \\ x_1 \in \{0,1,2\}, \\ x_2 \in \{0,1,2,3\}, \\ x_3 \in \{0,1,2,3,4\}, \\ x_4 \in \{0,1,2\}, \\ x_5 \in \{0,1,2,3,4,5\}, \\ x_6 \in \{0,1,2,3,4,5,6\}, \\ x_7 \in \{0,1,2,3\}, \\ x_8 \in \{0,1,2,3,4,5\}, \end{cases} \tag{2.8}$$

通过方向差商法求得的数据如表 2.1 所示。

表 2.1　8 处理机系统的最优分配

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
0	2	2	1	4	6	2	4
1	1	2	1	4	6	2	4
1	2	1	1	4	6	2	4
1	2	2	0	4	6	2	4
1	2	2	1	3	6	2	4
1	2	2	1	4	5	2	4
1	2	2	1	4	6	1	4
1	2	2	1	4	6	2	3

在可行域中，找到最小值为 8 的有 8 组。

（4）集中操作，结束。

当执行完分散的任务后，将每一个客户处理机的结果收集到服务器中。

2.1.3 算法分析

SQP 算法每次迭代都构造一个 QP 问题，且每次迭代的时间复杂度一样，因此只考虑一次迭代就可以了。

顺序执行时间为

$$t_s = m(2n+1) \tag{2.9}$$

顺序时间复杂性为 $O(n\times m)$ 。

对于并行优化而言，设处理机总数为 p，则从处理机有 p−1 个。

（1）通信。首先向从处理机广播仿真评估模型，并返回收到模型确认，即

$$t_{\text{conmm1}} = (p-1)(t_{\text{startup}} + t_{\text{data1}}) + t_{\text{startup}} + t_{\text{data0}} \tag{2.10}$$

式中，t_{startup} 是形成消息和启动传递所需的时间；t_{data0}、t_{data1} 是传递不同数据所需时间。

再执行分散操作，即

$$t_{\text{conmm2}} = (p-1)(t_{\text{startup}} + t_{\text{data2}}) \tag{2.11}$$

式中，t_{data2} 是分散操作需要的平均时间。

（2）计算。所有处理机并行地完成它们的仿真评估任务，即

$$t_{\text{comp}} = \frac{2nm}{p} = O\left(\frac{nm}{p}\right) \tag{2.12}$$

（3）通信。执行集中操作，即

$$t_{\text{conmm3}} = t_{\text{startup}} + t_{\text{data3}} \tag{2.13}$$

式中，t_{data3} 是集中操作需要的平均时间。

（4）总执行时间。总执行时间可表示为

$$t_p = \frac{2nm}{p} + 2pt_{\text{startup}} + (p-1)(t_{\text{data1}} + t_{\text{data2}}) + t_{\text{data3}} + t_{\text{data0}} \tag{2.14}$$

（5）加速系数。加速系数可表示为

$$a_{\text{index}} = \frac{t_s}{t_p} = \frac{2nm}{\dfrac{2nm}{p} + 2pt_{\text{startup}} + (p-1)(t_{\text{data1}} + t_{\text{data2}}) + t_{\text{data3}} + t_{\text{data0}}} \tag{2.15}$$

如果问题规模足够大，则加速系数可能接近 p。

（6）计算/通信比。计算时间与通信时间比可表示为

$$\text{cc} = \frac{2nm}{p[2pt_{\text{startup}} + (p-1)(t_{\text{data1}} + t_{\text{data2}}) + t_{\text{data3}} + t_{\text{data0}}]} \tag{2.16}$$

（7）并行效率。并行效率可表示为

$$\eta = \frac{t_s}{t_p p} = \frac{2nm}{2nm + p[2pt_{\text{startup}} + (p-1)(t_{\text{data1}} + t_{\text{data2}}) + t_{\text{data3}} + t_{\text{data0}}]} \tag{2.17}$$

如果问题规模足够大，则并行效率可接近 100%。

从算法分析中可以看出，通信开销占有相当的比例，t_{data0}、t_{data1}、t_{data3} 的大小和问题规模成正比增加。从加速系数可知，当问题有一定规模时，通信时间相对减少，加速系数增大。因此，并行优化算法是很有价值的。

2.2　动态响应并行优化过程

2.2.1　并行平台

并行系统拓扑结构如图 2.2 所示。此系统嵌入 CAD 中心的软件中。用户可以决定此系统是否要并行优化。Communication（Server）在服务器上先启动，然后启动客户端 Communication（Client）。并行仿真/优化时会自动查找设定的服务器端，建立连接。服务器有如下功能：

（1）RecieveTask();　　//接受从客户端来的任务申请（//：C++注释）

（2）RecieveFile();　　//接受 DLL 文件

（3）ExecuteParallelCalc();　//执行并行求解

（4）ReturnResult();　　//返回计算结果给客户端

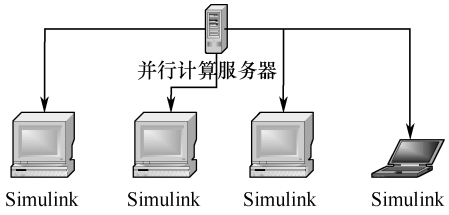


图 2.2　并行系统拓扑结构

并行平台有以下两个重要功能。

（1）多机并行冲突处理，即服务器忙，并行任务在执行 S；C_1，C_2，C_3，…，当 C_i 要进行计算时，首先判断服务器状态，若服务器忙，则本机进行单机运算，同时提交申请，服务器接收到申请，则在下一个循环中停止，再向该机器发送计算任务。

（2）并行优化可行性评估，即计算每次仿真/求解时间 A，与并行计算加上通信时间之和 B 比较，若 $A > B$，则采用并行处理，否则采用单机计算。若采用并行处理，则采用负荷均匀与能者多劳两种任务分配方式进行负载平衡。计算机利

用双线性障栅技术实现了进程同步，即主进程管理两个线性计数器，一个是任务计数器，另一个是结果计数器，只有当双线性计数器达到初始化时设置的数时，线程才能继续运行。

2.2.2 动态响应并行优化实现

并行执行流程如图 2.3 所示，其中 X 是设计变量，C 表示约束函数，O 表示目标函数，W 表示加权系数。并行执行流程中涉及若干关键过程，下面分别阐述。

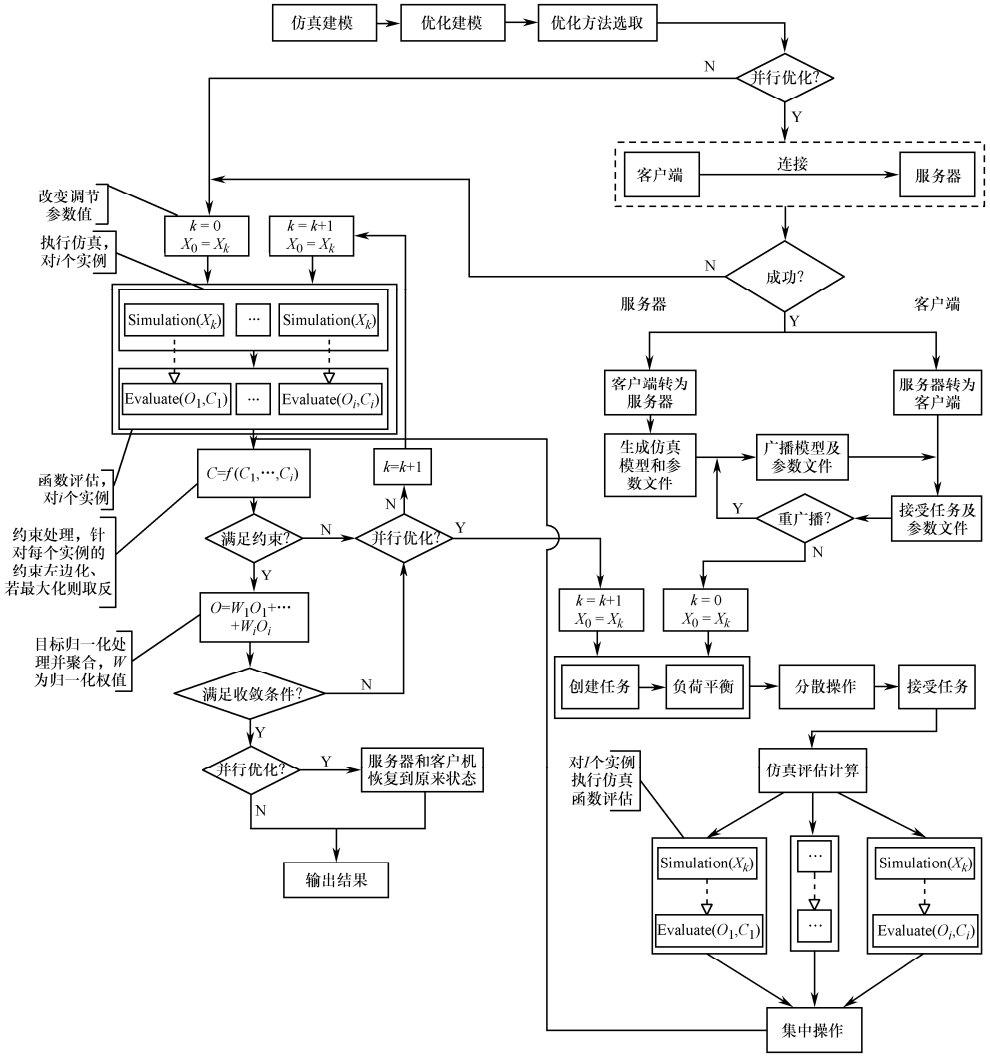


图 2.3 并行执行流程

1. 客户机和服务器转换角色

用户以其中一台装有 CAD 中心优化库软件的计算机为操作对象，将其转换成（或增加启动本机作为）服务器，如果原服务器空闲，则将所有现有的客户端与现在的服务器连接，包括在原服务器上添加一个执行器，共有执行器数 = 客户端个数+原服务器端；如果原服务器忙，则当前客户机执行单机运算。

2. 生成仿真模型与参数文件

编译仿真模型，生成仿真参数 mat 文件（MATLAB 数据格式文件），用于保存所有参数在各仿真步的参数值；将仿真模型生成执行文件 Simulator.dll，初始化时从仿真参数 mat 文件中读出参数值，这样每次迭代执行仿真时不需要重新生成仿真执行文件。

3. 客户端创建任务

若角色转换成功，首先创建并行任务存入队列（即向量容器 vector <SimulateData>，其中 SimulateData 是结构体），即如图 2.1 所示的工作池队列，然后分配任务创建执行器（即 map 容器，map<Socket *，vector<SimulateData>> mExecutors，其中 SimulateData 是结构体），使客户机和任务对应，便于管理。

4. 启动服务器任务分配与负载平衡程序，进行并行计算，计算结束通知客户端

5. 并行优化任务

并行优化任务为基类 Task（如下面的 C++伪代码），用于存储和设置每一个任务共有的信息。其他任务均从此基类派生。

```
class Task {                          //任务
protected:
int    m_uType;                       //命令类型
string m_sClientIP;                   //客户机名称
int    m_nPort;                       //端口号
bool  m_bFinished;                    //任务完成标志
char * m_sCommandLine;                //命令流，去报头
string m_sAppendMsg;                  //显示文本信
string m_sModifListStr;               //移除的列表信息
static CMySocket * m_pSocket;         //套接字
public:
Task();                               //构造函数
~Task();                              //析构函数
int    GetType();                     //获得命令类型
```

```
string GetClientIP();           //获得客户机名称
string GetAppendMsg();          //获得消息
void   GetListModStr(StringArray &a);  //获得移除信息
virtual bool Parse() = 0;       //解析命令，将字符流转化成对象内容
virtual string GenerateStream() = 0;   //将命令对象变成字符流
virtual bool Execute(CPtrList & pSockets) = 0; //执行命令
void   SetSocket(CMySocket* pSock);    //设置套接字
void   InitInfo(const string* sClient,int nPort,char* pBuf);
                                       //初始化
void   SetInfo(const string* sClient,int nPort,char* pBuf);
                                       //设置基本数据
……};
```

几个关键任务包括初始化仿真器任务（InitSimulateDLLTask）、初始化参数文件任务（InitParaMatTask）、执行仿真任务（SimulateRunTask）和仿真结果任务（SimulateResultTask），其中 InitSimulateDLLTask 用于存储初始化模型动态库文件信息；InitParaMatTask 用于存储初始化参数文件信息；SimulateRunTask 用于执行仿真任务，读取初始化参数文件，仿真评估结束后将结果写入结果数据；SimulateResultTask 用于存储结果数据。

6. 集中操作

服务器收集客户机的仿真结果，并保存在仿真结果 mat 文件中。由仿真结果 mat 文件来计算目标或约束变量值，对于未激活的不考虑；变量值需要根据其取值类型（如最终值等），通过仿真时间变化曲线来确定。

7. 目标聚合

对于多实例、多目标问题，设 q 为所有的归一化目标值 $q_{ij}=\dfrac{c_{ij}}{d_{ij}},i,j\in S_{\min}$ 组成的向量，其中 $S_{\min}$ 表示所有实例中最小化的实例目标集合，q_{ij}、c_{ij}、d_{ij} 分别表示第 j 个实例的第 i 个目标的归一值、目标值和期望值。采用向量最大范式，优化的总目标就是获得一个 q 中最大值最小的归一向量所对应的调节参数集合的值。

令 $\alpha:=\max\{q_{ij}\},i,j\in S_{\min}$，这样如果 $\alpha_1<\alpha_2$，则认为第一个设计比第二个设计更满意；如果 $a\leqslant 1$，则该设计是满足约束的，因为此时各个目标值都小于限定的约束权值。

按照上述描述，优化问题可认为是寻找最小的 α，即

$$\alpha^*=\min\{\alpha\}$$

因此，该多实例优化问题的数学模型描述如下：

$$\begin{cases} \min\limits_{t} \max\limits_{i,j\in S_{\min}} \left[\dfrac{c_{ij}(t)}{d_{ij}} \right] \\ \text{s.t.} \quad c_{ij}(t) \leqslant d_{ij}, i,j \in I \\ \qquad c_{ij}(t) = d_{ij}, i,j \in E \\ \qquad t_{\min} \leqslant t \leqslant t_{\max} \end{cases} \tag{2.18}$$

式中，t 表示时间，$t_{\min}$ 和 $t_{\max}$ 分别表示仿真时间的上、下限。

2.3　实例分析

下面以 F14 战机简易模型的控制系统参数优化为例（如图 2.4 所示），说明多实例、多目标问题仿真优化的并行求解。该模型用于对战机模型的纵向运动响应进行仿真与分析，优化设计的目的是确定合适的控制参数，使该响应符合操控要求。MWorks 软件提供了建立优化模型的统一界面（如图 2.5 所示）。用户既可从编译仿真模型后得到的参数变量结构树中选择并设置优化模型信息，也可读取保存在仿真模型中的优化信息，在此基础上构建新的优化模型。

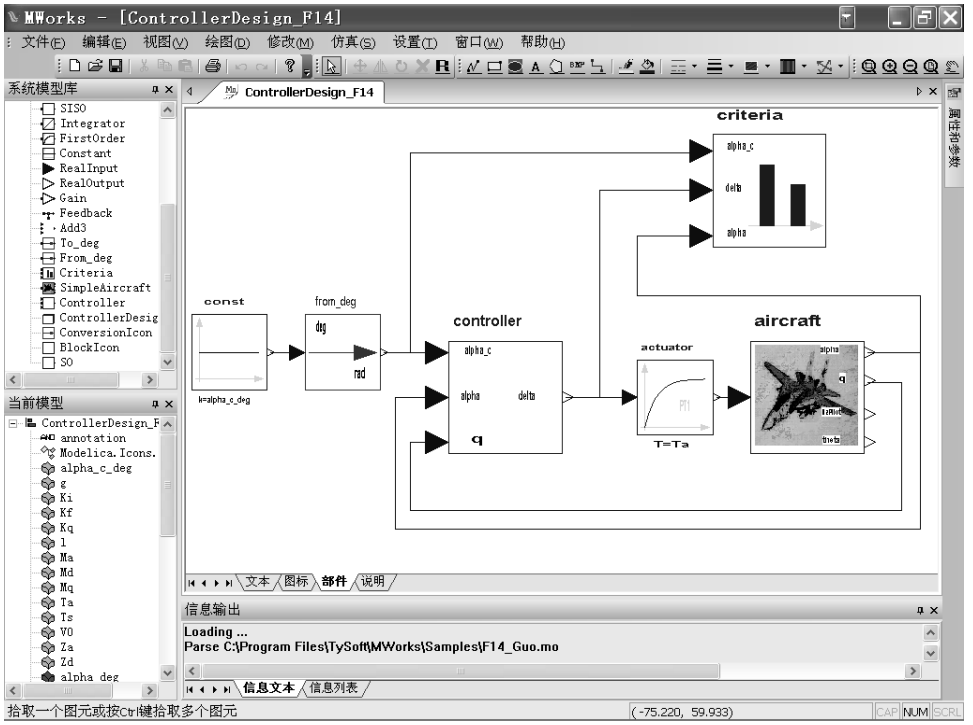


图 2.4　F14 战机简易模型的控制系统参数优化

图 2.5　构建多实例、多目标优化模型的统一界面（一）

1. 设计参数

选择 3 个控制器增益作为设计参数，并设置初始值和边界值（如图 2.6 所示），即 $K_f \in [-10,-0.5]$，初始值取−6；$K_i \in [-10,-0.5]$，初始值取−2；$K_q \in [0.1,10]$，初始值取 0.5。设计变量的意义如下。

（1）K_f：攻击角的比例增益。

（2）K_i：攻击角的积分增益。

（3）K_q：俯仰速率的比例增益。

图 2.6　构建多实例、多目标优化模型的统一界面（二）

2. 模型实例

选取战机模型的 5 个空气动力学参数作为实例参数，并定义了如表 2.2 所示

的两个模型实例，其中实例 2 的参数值是在实例 1 的参数值基础上变动+10%后得到的。该多实例优化的目的是在战机模型参数扰动的情况下仍能获得满意的优化结果。实例参数的意义如下。

（1）M_a：俯仰角对攻击角的变化系数。

（2）M_d：俯仰扭矩对单位攻击角偏差的变化系数。

（3）M_q：俯仰扭矩对俯仰速率的变化系数。

（4）Z_a：单位攻击角的提升系数。

（5）Z_d：单位攻击角偏差的提升系数。

表 2.2　模型实例

实例	M_a	M_d	M_q	Z_a	Z_d
1	−5	−7.5	−0.7	−0.67	−0.2
2	−4.5	−6.75	−0.63	−0.603	−0.18

3. 实例目标和约束

选择攻击角的超调量（f_1）、上升时间（f_2）和调节时间（f_3）作为实例目标，以攻击角最大偏差（c）为约束，并设定各实例的期望值，如表 2.3 所示。

表 2.3　实例目标和约束的期望值

实例	f_1	f_2	f_3	c
1	0.01	0.5	4	2
2	0.01	0.5	4	2

4. 优化模型数学表达式

建立以上优化模型后可以得到如下优化模型的数学表达式：

$$\begin{cases} \text{find } [K_f, K_i, K_q] \\ \min \sum \left| \dfrac{f_{ij}}{d_{ij}} \right| \\ \text{s.t. } c_{ij} \leqslant 2 \\ \quad -10 \leqslant K_f \leqslant -0.5 \\ \quad -10 \leqslant K_i \leqslant -0.5 \\ \quad 0.1 \leqslant K_q \leqslant 10.0, \left(i = \{1,2,3\}, j = \{1,2\}\right) \end{cases} \tag{2.19}$$

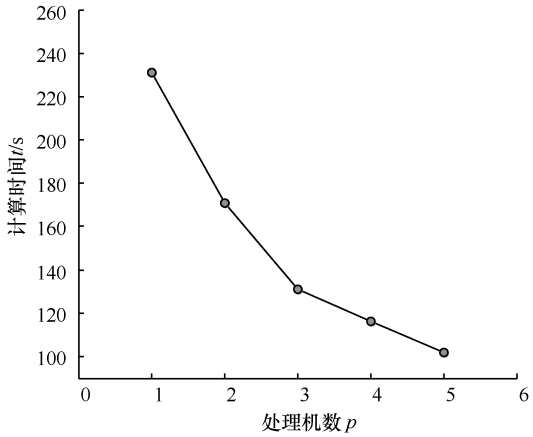
5. 计算机性能参数

利用 2 节点机群系统求解。单机及机群系统的机器配置为：Pentium(R) D CPU，3.0GHz，1.00GB RAM。网络环境为 100M 局域网。

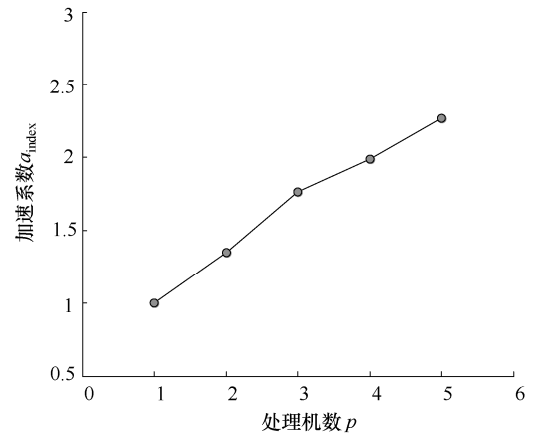
分别采用序列二次规划法串行和并行求解式（2.19），经过 15 次迭代后收敛，优化结果为：$[K_f, K_i, K_q] = [-3.0432, -3.3956, 0.7723]$。优化求解后得到的目标和约束值如表 2.4 所示。从表 2.4 中可以看出所有的目标和约束均满足表 2.3 设定的实例期望。单机耗时 339s，机群耗时 249s，可以看出 2 节点机群的计算时间有明显下降。下面利用 3、4、5 节点机群系统分别求解，得出如图 2.7 所示的曲线，由图可知，计算节点增多，计算时间减小，加速比增大，但是并行效率有所下降。

表 2.4　优化求解后得到的目标和约束值

实例	f_1	f_2	f_3	c
1	0.0000	0.4269	3.4150	1.9380
2	0.0085	0.4140	3.2543	2.0000

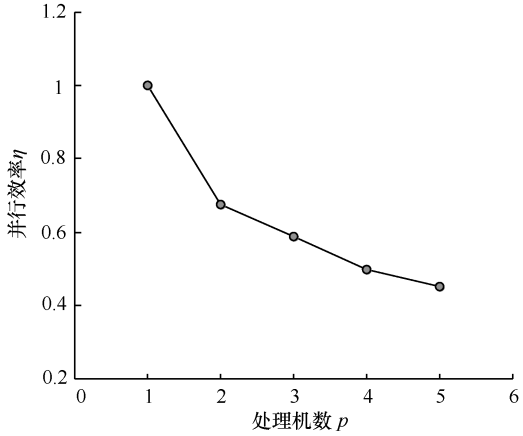


(a) 处理机数与计算时间的关系



(b) 处理机数与加速比的关系

图 2.7　处理机数与计算时间、加速比、并行效率的关系



(c) 处理机数与并行效率的关系

图 2.7　处理机数与计算时间、加速比、并行效率的关系（续）

在 2 节点机群系统中，求解 F14 战机算例的加速比和并行效率，如下所示。

加速比：$a_{\text{index}}=\dfrac{t_s}{t_p}=\dfrac{339}{249}=1.36$

并行效率：$\eta=\dfrac{t_s}{t_p p}=\dfrac{339}{249\times 2}=68.07\%$

根据 Amdahl 定律知

加速极限：$a_{\text{limit}}=\dfrac{1}{\dfrac{t_s-t_p}{t_s}}=\dfrac{1}{\dfrac{339-249}{339}}=3.77$

由此说明加速比的极限远没有达到。可以通过增加处理器的数量来使加速比进一步提高。实际上，仅单项仿真评估任务并行处理和梯度计算并行化，就可得到比较好的效率。若提高机群性能，改善网络环境等硬件设备，或者改进算法，将进一步提高计算效率。

本书作者从 SQP 优化算法的并行化及实现等方面研究了 SQP 并行参数优化设计方法。研究表明在 SQP 算法中，梯度计算和模型仿真评估并行化的效果较好，明显加速了优化设计的进度，在基于多学科仿真的多实例、多目标优化中体现得更加明显；SQP 算法成功与否依赖于梯度计算的准确性和有效性；利用机群系统计算可用较低的成本获取较大的计算能力，解决仿真优化中高耗时的问题。MWorks 系统可以很容易地引入那些可买到的最新的处理器，扩充机群系统。对于 SQP 并行算法的进一步研究，应考虑全局并行化和线性搜索并行化，以及全局、函数评估、梯度计算与线性搜索混合并行化和伴随梯度法[49、50]。

2.4 本章小结

本章研究了基于仿真模型的动态响应优化中 SQP 算法的并行处理与调度策略，提出了基于仿真模型的 SQP 并行优化问题中的抽象调度模型，即等式约束离散变量优化模型，从算法理论上深入分析了其可行性；采用机群系统构建了并行仿真优化环境，在 CAD 中心的软件平台下开发了并行优化模块；以 F14 战斗机简易模型的控制参数动态响应优化为例，验证了本章所介绍方法的有效性。

第 3 章　基于时间谱元法的动态响应优化算法

对于可建立精确物理模型的机械结构，在动态载荷作用下为使机器动态性能指标达到最优，可以建立动态响应方程，然后进行动态响应优化。当然，并行化可以起到一定作用，但只是从如何减少耗时方面考虑的，然而仿真模型的精度给动态响应优化带来了困难，因此对于这类问题，非常有必要研究一种高精度、高效率求解动态响应的技术。

本章通过应用时间谱元法来有效优化激励动态系统。此方法将时间响应转化为显式代数形式，并比较了在 GLL 点执行约束与在响应的局部极值点执行约束的优化成本，前者可以利用谱元法的解，后者可应用一维线性搜索。

3.1　谱元法

从理论上分析，在一定点数上插值，当这些点是对应的正交多项式的零点时获得的插值精度最高[59]。如图 3.1 所示，均匀点分布和 GLL 点分布分别近似于

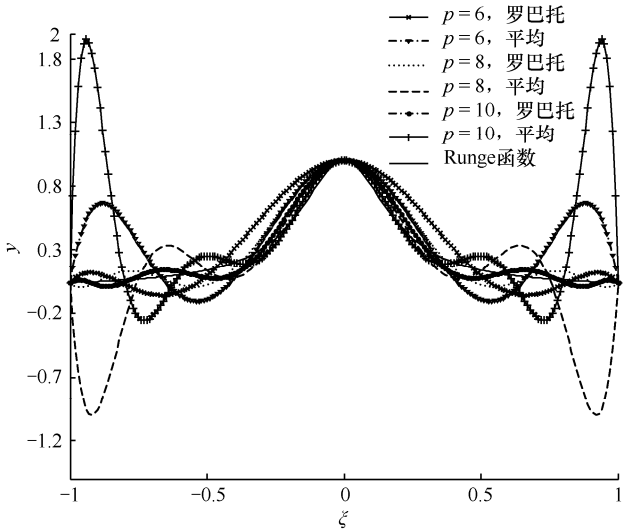


图 3.1　均匀点和 GLL 点的插值近似 Runge 函数

Runge 函数式（3.1）。使用均匀分布的有限元法，随着插值次数的增加，近似的数值误差极大或近似失败；而 GLL 插值有明显的优势。GLL 点在标准区间的分布如图 3.2 所示。

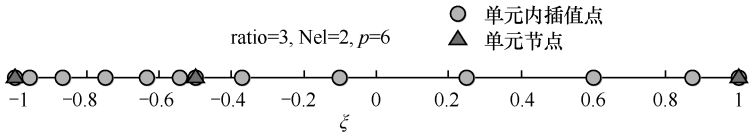


图 3.2　GLL 点的分布

$$f(x)=\frac{1}{1+25x^2} \tag{3.1}$$

微分方程常用来描述自然界的一些物理现象，包括动力学方程或方程组。差分法是直接基于这些微分方程或方程组的离散化，但是在多数情况下，描述同一物理过程或现象可采用不同的形式。从物理上的守恒定律出发可以导出变分原理，而变分问题与微分方程定解问题在某种意义下等价。谱元法是基于变分原理的一种离散计算方法。下面讨论一阶线性微分方程的初值问题。高阶线性微分方程的初值问题可以转化成一阶线性微分方程组。

$$\left.\begin{aligned}&\frac{\mathrm{d}x}{\mathrm{d}t}+\boldsymbol{A}_s x=f(x,t)\\&x(0)=x_0\end{aligned}\right\} \tag{3.2}$$

式中，x 是依赖于时间的状态变量，且 $x\in R^{N_v}$，N_v 为状态变量个数；t 是时间，$f(x,t)$ 是状态变量 x 和时间 t 的函数；$\boldsymbol{A}_s$ 表示状态变量的耦合矩阵，和时间无关。

在上两图中，ratio 表示最后一个单元与第一个单元的比值；Nel 表示单元数；p 表示单元内插值节点数。

3.1.1　瞬态和稳态分析

应用谱元法将每一个状态变量在给定时间段内离散化，近似为 m 次 Lagrange 多项式：

$$\tilde{x}^{(j)}(\xi)=\sum_{k=0}^{m}x^{(j)}(\xi_k)P_k^{(j)}(\xi) \tag{3.3}$$

式中，$P_k^{(j)}(\xi)$ 为第 j 个单元的第 k 个 m 次 Lagrange 多项式；ξ_k 为定义在[−1，1]上的 GLL 点；$x^{(j)}(\xi_k)$ 是第 j 个单元上未知节点在 GLL 点的值。Lobatto 多项式是 Legendre 多项式的微分定义的正交多项式（见文献[59]，p146，p151）。式中的 $\xi\in[-1,1]$，是经过式（3.4）映射获得的：

$$x(\xi)=\frac{1}{2}(x_2^{(j)}+x_1^{(j)})+\frac{1}{2}(x_2^{(j)}-x_1^{(j)})\xi \tag{3.4}$$

式中，$x_1^{(j)}$、$x_2^{(j)}$ 分别为第 j 个单元的第一个节点和第二个节点；ξ 是经过域变换后的变量，且 $\xi \in [-1,1]$。这个域如图 3.3 所示。

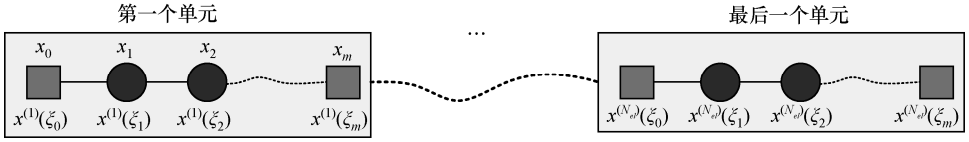


图 3.3　时间域离散为各谱单元（每一个单元基于 GLL 点的 m 次 Lagrange 多项式）

将式（3.3）代入式（3.2），在每一个单元上应用布勃诺夫−伽辽金（Bubnov-Galerkin）法[60]使得插值误差最小，得出：

$$\sum_{j=1}^{\mathrm{Nel}} \int_{-1}^{1} P_n^{(j)} \left[\frac{\mathrm{d}\tilde{x}^{(j)}}{\mathrm{d}\xi} + \frac{h^{(j)}}{2} \{ A_s \tilde{x}^{(j)} - f^{(j)}(\tilde{x}^{(j)}, \xi) \} \right] \mathrm{d}\xi = 0 \tag{3.5}$$

式中，$n = 0,1,\cdots,m$；$h^{(j)}$ 是第 j 个单元的长度；$P_n^{(j)}$ 是 m 次 Lagrange 多项式的第 n 个插值基函数。

对式（3.5）分部积分得出：

$$\sum_{j=1}^{N_{el}} [\tilde{x}^{(j)} P_n^{(j)} \,|_{-1}^{1} - \int_{-1}^{1} (\mathrm{d}\tilde{x}^{(j)} \frac{\mathrm{d}P_n^{(j)}}{\mathrm{d}\xi} + \frac{h^{(j)}}{2} \{ A_s \tilde{x}^{(j)} P_n^{(j)} - f^{(j)}(\tilde{x}^{(j)}, \xi) P_n^{(j)} \}) \mathrm{d}\xi] = 0 \tag{3.6}$$

式（3.6）中的积分用 Gauss-Lobatto 求积公式求得，其公式为

$$\int_{-1}^{1} I \mathrm{d}\xi = \sum_{q=0}^{m} I(\xi_q) \omega_q \tag{3.7}$$

式中，I 是 ξ 的一般函数；ω_q 是 Gauss-Lobatto 积分在 q 点的权值[60]。将每一个单元用矩阵形式表示为

$$\boldsymbol{\Phi} \begin{Bmatrix} x(\xi_0) \\ x(\xi_1) \\ \vdots \\ x(\xi_m) \end{Bmatrix}^{(j)} = \boldsymbol{A}_s \boldsymbol{I}_{\omega}^{(j)} \begin{Bmatrix} x(\xi_0) \\ x(\xi_1) \\ \vdots \\ x(\xi_m) \end{Bmatrix}^{(j)} - \boldsymbol{I}_{\omega}^{(j)} \boldsymbol{f}^{(j)} \tag{3.8}$$

其中，

$$\boldsymbol{\Phi} = \begin{bmatrix} \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_0} \omega_0 + 1 & \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_1} \omega_1 & \cdots & \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_m} \omega_m \\ \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_0} \omega_0 & \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_1} \omega_1 & \cdots & \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_m} \omega_m \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_0} \omega_0 & \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_1} \omega_1 & \cdots & \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_m} \omega_m - 1 \end{bmatrix} \tag{3.9}$$

$$\boldsymbol{I}_{\omega}=\frac{h^{(j)}}{2}\begin{bmatrix}\omega_0 & 0 & \cdots & 0\\ 0 & \omega_1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \omega_m\end{bmatrix} \tag{3.10}$$

由于相邻两个单元共享其中一个元素，所以应满足：

$$x^{(j)}(\xi_m)=x^{(j+1)}(\xi_0) \tag{3.11}$$

接着通过式（3.8），利用连接矩阵[59]$\boldsymbol{C}$，将一个状态变量的所有谱单元组装起来，得到一个状态变量的 Galerkin 近似方程组：

$$\boldsymbol{B}_u\boldsymbol{X}_u=\boldsymbol{A}_s\boldsymbol{B}_{\omega}\boldsymbol{X}_u-\boldsymbol{B}_{\omega}\boldsymbol{F}(X_u) \tag{3.12}$$

式中，$\boldsymbol{B}_u$、$\boldsymbol{B}_{\omega}$是全局微分矩阵与全局权矩阵；$\boldsymbol{F}(X_u)$是激励力的全局形式，其中

$$\boldsymbol{X}_u=[x|_{t_0}\quad x|_{t_1}\cdots x|_{t_m\times\mathrm{Nel}+1}] \tag{3.13}$$

它是用谱元法表示的状态变量 x 的所有时间节点变量。初始条件处理为将 $\boldsymbol{B}_u$ 第一行和第一列的第一个元素赋为 1，$\boldsymbol{B}_u$ 的第一行和第一列的其余元素赋为零；再将 $\boldsymbol{B}_{\omega}$的第一个元素赋为零，同时将 $-\boldsymbol{B}_{\omega}\boldsymbol{F}(X_u)$ 的第一个元素赋为式（3.2）中所示的初始值，即 $x|_{t=0}=x_0$。

稳态响应分析和瞬态响应分析的谱离散相同，而组装全局微分矩阵和全局权矩阵时不同。由于初始条件不影响稳态响应，所以没有初始条件处理这一步。由于周期的特殊性，要求第一个单元的第一个节点和最后一个单元的最后节点相等，即

$$s^{(1)}(\xi_0)=x^{(\mathrm{Nel})}(\xi_m) \tag{3.14}$$

处理第一行、第一列和最后一行、最后一列。将全局微分矩阵 $\boldsymbol{B}_u$ 的最后一行加到第一行上，最后一列加到第一列上，然后去掉最后一行和最后一列；对全局权矩阵 $\boldsymbol{B}_{\omega}$也做同样的处理；对于 $\boldsymbol{F}(X_u)$，将最后一个元素加到第一个元素上，去掉最后一个元素。这样，$\boldsymbol{X}_u$ 变成：

$$\boldsymbol{X}_u=\left[x\Big|_{t_0}\quad x\Big|_{t_1}\quad\cdots\quad x\Big|_{t_m\times\mathrm{Nel}}\right]^{\mathrm{T}} \tag{3.15}$$

3.1.2 全部状态变量的全局组装和求解

对于 N_v 个状态变量，通过耦合矩阵 $\boldsymbol{A}_s$（$N_v\times N_v$ 的方阵）的张量叉乘得到全部状态变量的全局组装式（3.16）：

$$(\boldsymbol{I}\otimes\boldsymbol{B}_u)\boldsymbol{X}_{ug}=(\boldsymbol{A}_s\otimes\boldsymbol{B}_{\omega})\boldsymbol{X}_{ug}-(\boldsymbol{I}\otimes\boldsymbol{B}_{\omega})\boldsymbol{F}_{ug}(\boldsymbol{X}_{ug}) \tag{3.16}$$

式中，$\boldsymbol{I}$ 是 $N_v\times N_v$ 单位矩阵；$\boldsymbol{X}_{ug}$ 是所有状态变量在时间节点的集合。

对于瞬态响应 $\boldsymbol{X}_{ug}$ 而言，有

$$\boldsymbol{X}_u=\left[\begin{Bmatrix} x_1\big|_{t_0} \\ x_1\big|_{t_1} \\ \vdots \\ x_1\big|_{t_m\times \mathrm{Nel}+1} \end{Bmatrix} \cdots \begin{Bmatrix} x_{\mathrm{Nel}}\big|_{t_0} \\ x_{\mathrm{Nel}}\big|_{t_1} \\ \\ x_{\mathrm{Nel}}\big|_{t_m\times \mathrm{Nel}+1} \end{Bmatrix}\right]^{\mathrm{T}} \tag{3.17}$$

化简式（3.16）得到：

$$\boldsymbol{G}\boldsymbol{X}_{ug}=-\boldsymbol{B}_{\omega g}\boldsymbol{F}_{ug}(\boldsymbol{X}_{ug}) \tag{3.18}$$

$$\boldsymbol{G}=\boldsymbol{B}_{ug}-\boldsymbol{A}_{ug} \tag{3.19}$$

3.2　逐步时间谱元法

结构动态响应问题在空间经有限元离散以后得到二阶线性常微分方程组。目前，各种文献中提出了许多常微分方程的数值求解方法，如中心差分法、Newmark、Wilson-θ 、Houbolt 法等，它们都是在时间域上的差分法。在计算结构动态响应的数值方法中，差分法占据重要地位。对于该方法，采用足够小的时间步长计算二阶微分方程或方程组，直到收敛，计算的稳定性和准确性取决于时间步的大小，这妨碍了计算效率。对于受脉冲激励的系统而言，应用基于频率的方法求解结果的精度不高。对于脉冲激励的动态系统，谱元法在半连续附近采用 h 型精细方案，并且将微分方程或方程组转化为代数方程组，且在不增加单元数的前提下达到谱收敛[1]。文献[2]深入探讨了在时间域内离散动态响应，将运动微分方程组转化成代数方程组，精确解出瞬态响应。用高斯罗巴托勒让德（Gauss-Lobatto-Legendre，GLL）点法和关键点法处理时间约束，其核心问题是高效率求解动态响应方程和有效处理约束。以最简单的多自由度动力学设计问题——两自由度弹簧减振器设计为例，尽管它可以获得非常满意的结果，但是效率成为其工程应用的致命点。可以看出，仅仅两自由度系统，时间划分为 6 个单元，每一个单元进行 16 次插值，用 Intel 奔腾双核 E5800 CPU（3200MHz），1GB DDR3（1333MHz）内存的台式计算机仿真一次平均需要 70s 左右，然而工程问题经有限元离散后自由度数非常大，仿真耗时将是不可想象的。

Patera 在 1984 年提出的谱元法，从理论上分析，它是在一定点数上插值的，当这些点对应正交多项式的零点时获得的插值精度最高[2、3]。应用空间-时间离散整个时间域是由 Oden[5]，Fried[6]，Argyris 等[7]首先提出的，这种离散基于哈密尔顿动力学原理，它应用哈密尔顿原理对动态载荷系统进行时间近似[8]，利用余量

加权法获得了与时间近似相同的结果[9]。Bar-Yoseph[10]比较了这两种方法。本节提出逐步时间谱元法，将仿真时间分段，在每一段内再划分单元，对每个单元进行插值，由于仿真时间段可以划分得很小，这样在每一个时间段内单元数很少就能达到很高精度，进而可大大提高动态响应分析的效率。

3.2.1 线性动力学方程的逐步时间谱元法

逐步时间谱元法[61]的主要步骤：

（1）首先转化二阶微分方程组为一阶微分方程组，应用 Bubnov-Galerkin 原理转化为等效的积分形式，代入单元积分表达式，得到单元谱元方程；

（2）将仿真时间 $t\in[t_0,t_n]$ 分段，划分为 $[t_0,t_1]$，$[t_1,t_2]$，$[t_2,t_3]$，…，$[t_{n-1},t_n]$；

（3）将一个时间段划分成许多单元，每个单元由若干个节点组成；

（4）用 Legendre 正交多项式作为基函数，将每一个单元的近似解表达为基函数的线性组合；

（5）将一个分段中的所有单元的谱元法按一定规则合成其一段时间内的总体谱元方程；

（6）解一段时间的总体谱元方程，得到其分段内的全局近似解；

（7）对于第一个单元 $[t_0,t_1]$，初始值就是运动微分方程的初始条件，而从第二个时间段开始，其初始条件是前一个时间段右端时间点处谱元法的解，以此类推，重复（2），（3），（4），（5），（6）直到所有分段；

（8）最后将每个时间段的时间点连接成为全局时间点，以及将相对应的各响应连接成为最终解。

3.2.2 线性结构动态响应方程及其转化形式

线性结构动态响应方程可表达为

$$\boldsymbol{M}\ddot{x}+\boldsymbol{C}\dot{x}+\boldsymbol{K}x=\boldsymbol{F} \tag{3.20}$$

式中，$\boldsymbol{M}$ 为质量矩阵；$\boldsymbol{C}$ 为阻尼矩阵；$\boldsymbol{K}$ 为刚度矩阵；$\boldsymbol{F}$ 为时间函数载荷向量；x 为位移向量；$\ddot{x}$ 为加速度向量；$\dot{x}$ 为速度向量，初始条件为 $x(0)=b_0$，$\dot{x}(0)=v_0$；$\boldsymbol{M}$，$\boldsymbol{C}$，$\boldsymbol{K}$ 不随时间变化；x，$\ddot{x}$，$\dot{x}$ 是时间的函数；$\boldsymbol{F}$ 是时间的任意函数，$t\in[t_0,t_n]$。

在时间谱元法中，为了用数值方法解动态响应方程，可通过设 $x_1=\dot{x}$，$x_2=x$ 将式（3.20）转化为一阶线性常微分方程组式（3.21），即

$$\begin{Bmatrix}\dot{x}_1\\ \dot{x}_2\end{Bmatrix}+\begin{bmatrix}\boldsymbol{M} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{C} & \boldsymbol{K}\\ -\boldsymbol{I} & 0\end{bmatrix}\begin{Bmatrix}x_1\\ x_2\end{Bmatrix}=\begin{bmatrix}\boldsymbol{M} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}^{-1}\begin{Bmatrix}F\\ 0\end{Bmatrix} \tag{3.21}$$

$$\{x_1(0)\}=\{x_1^0\},\{x_2(0)\}=\{x_2^0\}$$

3.2.3　时间分段及单元划分

如图 3.4 所示，将仿真时间 $t\in[t_0,t_n]$ 分割为 n 个互不相交的子区间段，即 $[t_0,t_1]$，$[t_1,t_2]$，$[t_2,t_3]$，…，$[t_{n-1},t_n]$，对每一段时间划分若干单元，每个单元配置若干个点。当进行时间分段时，可以根据动态载荷的变化情况，在载荷突变的地方将时间段划分得尽量小，在载荷平缓的地方将时间段划分得适当大一点，这样能保证在动态载荷的所有地方都有足够的精度。

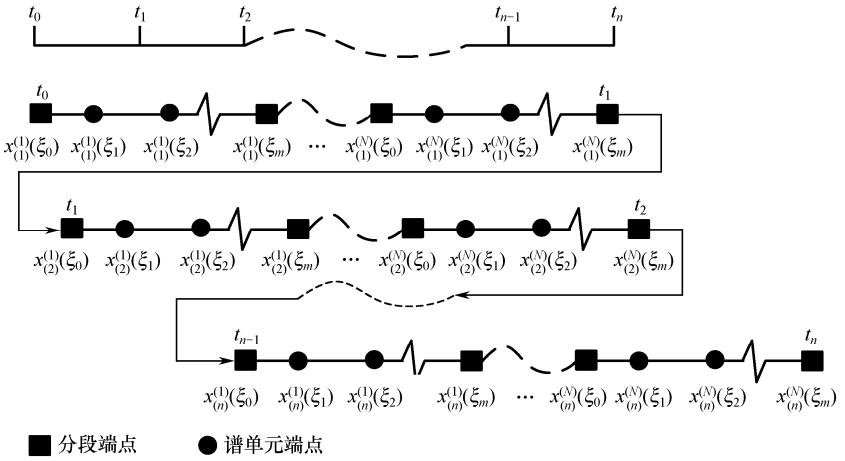


图 3.4　逐步时间谱元法离散时间

3.2.4　单元分析

区间分解以后，为提高近似解的精度，可以在单元区间上再增加插值点，以此增加近似解的阶次，这在有限元里被称为高阶元，即 hp 方法。如何选择增加节点的位置，如何选择形函数并通过插值来得到近似解，将时间谱元法从普通的高阶有限元法中区分开来呢？

在时间谱元法中，为了达到谱方法的收敛速度，并且增加数值计算的精度，单元内插值点应放置在特殊的配置点上，它们是由 N+1 个 Legendre–Gauss-Lobbatto 点构成的，而基函数由有限项正交多项式的和来表示，构成单元上任意配置点的形函数，如此可在有限的插值点上获得指数级的收敛速度。

在每一个单元上，将定义在插值点上的形函数通过 Lagrange 插值得到近似函数。换句话说，任意一个定义在参考单元上的函数 $\tilde{x}^{(j)}(\xi)$，都可以用下式近似：

$$\tilde{x}^{(j)}(\xi)=\sum_{k=0}^{m}x^{(j)}(\xi_k)P_k^{(j)}(\xi) \tag{3.22}$$

式中，$P_k^{(j)}(\xi)$ 为 j 单元的 k 次 Lagrange 多项式；ξ_k 为定义在[−1，1]上的 GLL 点；$x^{(j)}(\xi_k)$ 是单元 j 上未知节点在 GLL 点的值。

有两种类型的正交多项式可用于时间谱元法中：Chebyshev 和 Legendre 正交多项式。这里只讨论 Legendre 展开。

勒让德多项式是 Sturm-Liouville form 方程的解：

$$[(1-\xi^2)P'(\xi)]' + n(n+1)P(\xi) = 0 \tag{3.23}$$

它的权函数 $\omega = 1$。k 阶 Legendre 多项式可以定义如下：

$$P_k(\xi) = \frac{1}{2^k k!}\frac{\mathrm{d}^k}{\mathrm{d}\xi^k}[(\xi^2-1)^k] \tag{3.24}$$

由于 Legendre 多项式的零点不包括区间端点，所以引入 Lobbato 多项式。Lobbato 多项式是通过 Legendre 多项式的微分定义的[59]，即

$$L_{o_k}(\xi) = P'_{k+1}(\xi) \tag{3.25}$$

它满足正交特性。

Gauss-Lobatto-Legendre 点及其权重定义如下：

$$(1-\xi^2)P'_N(\xi) = 0 \tag{3.26}$$

式（3.26）的解为 GLL 点，其中 N 为插值次数。

将每一个状态变量在给定时间段内离散化，近似为 m 次 Lagrange 多项式［见式（3.22）］。在每一个单元上应用 Bubnov-Galerkin 法[60]使得插值误差最小，由此得出：

$$\sum_{j=1}^{\mathrm{Nel}}\int_{-1}^{1} P_n^{(j)}\left[\frac{\mathrm{d}\tilde{x}^{(j)}}{\mathrm{d}\xi} + \frac{h^{(j)}}{2}\{A_s\tilde{x}^{(j)} - f^{(j)}(\tilde{x}^{(j)},\xi)\}\right]\mathrm{d}\xi = 0 \tag{3.27}$$

将每一个单元用矩阵形式表示为

$$L^e x^e(t) = F^e(t) \tag{3.28}$$

式中，$L^e = \Phi - A_s I_\omega^{(e)}$；$x^e(t) = ([x(\xi_0)\ x(\xi_1)\ \cdots\ x(\xi_m)]^{(e)})^{\mathrm{T}}$；$F^e(t) = -I_\omega^{(e)} f^{(e)}$，$I_\omega$ 为 ξ 的一般函数，I_ω 满足 $\int_{-1}^{1} I\mathrm{d}\varepsilon = \sum_{q=0}^{m} I(\varepsilon_q)\omega_q$，$\omega_q$ 是 Gauss-Lobatto 积分在 q 点的权值。

$$\Phi = \begin{bmatrix} \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_0}\omega_0 + 1 & \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_1}\omega_1 & \cdots & \dfrac{\mathrm{d}P_0}{\mathrm{d}\xi}\Big|_{\xi_m}\omega_m \\ \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_0}\omega_0 & \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_1}\omega_1 & \cdots & \dfrac{\mathrm{d}P_1}{\mathrm{d}\xi}\Big|_{\xi_m}\omega_m \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_0}\omega_0 & \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_1}\omega_1 & \cdots & \dfrac{\mathrm{d}P_m}{\mathrm{d}\xi}\Big|_{\xi_m}\omega_m - 1 \end{bmatrix}, \quad I_\omega = \frac{h^{(j)}}{2}\begin{bmatrix} \omega_0 & 0 & \cdots & 0 \\ 0 & \omega_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \omega_m \end{bmatrix}$$

3.2.5　集成总体时间谱元方程及求解

总体合成的任务是将所有单元谱元方程按序叠加，得到总体谱元方程，即一个时域上的线性方程组。对于 N_v 个状态变量，通过耦合矩阵 $\boldsymbol{A}_s$（$N_v \times N_v$ 的方阵）的张量叉乘得到全部状态变量的全局组装式（3.29），即

$$(\boldsymbol{I} \otimes \boldsymbol{B}_u)\boldsymbol{X}_{ug} = (\boldsymbol{A}_s \otimes \boldsymbol{B}_\omega)\boldsymbol{X}_{ug} - (\boldsymbol{I} \otimes \boldsymbol{B}_\omega)\boldsymbol{F}_{ug}(\boldsymbol{X}_{ug}) \tag{3.29}$$

式中，$\boldsymbol{B}_u$、$\boldsymbol{B}_\omega$ 是全局微分矩阵与全局权矩阵；$\boldsymbol{F}_{ug}(\boldsymbol{X}_{ug})$ 是激励力的全局形式；$\boldsymbol{I}$ 是 $N_v \times N_v$ 单位矩阵；$\boldsymbol{X}_{ug}$ 是所有状态变量在时间节点的集合。

化简式（3.29）得到：

$$\boldsymbol{G}\boldsymbol{X}_{ug} = -\boldsymbol{B}_{\omega g}\boldsymbol{F}_{ug}(\boldsymbol{X}_{ug}) \tag{3.30}$$

式中，$\boldsymbol{G} = \boldsymbol{B}_{ug} - \boldsymbol{A}_{ug}$；未知向量 $\boldsymbol{X}_{ug}$ 包含了时间段的所有单元中所有 GLL 配置点上的位移和速度离散解；$\boldsymbol{G}$ 是时间段的全局线性矩阵。

式（3.30）所示的线性代数方程组可以直接求解。

3.3　初始条件

第一个时间段 $[t_0, t_1]$ 的初始条件就是结构动态响应方程的初始条件。从第二个时间段 $[t_1, t_2]$ 开始，直到最后一个时间段 $[t_{n-1}, t_n]$，每一个时间段的初始条件为其前面一个时间段的末端点的离散解，即

$$\begin{aligned}
&\left\{x_1^{(0)}(0)\right\} = \{x_1^0\}, \left\{x_2^{(0)}(0)\right\} = \{x_2^0\} \\
&\left\{x_1^{(i)}(t_i)\right\} = \{x_N^{(i-1)}(t_i)\}, \{x_2^{(i)}(t_i)\} = \{x_{2N}^{(i-1)}(t_i)\} \\
&i = 1, 2, \cdots, n
\end{aligned} \tag{3.31}$$

3.4　基于时间谱元法的动态响应优化

机械结构动态响应优化设计问题的目标为选择设计变量向量 $\boldsymbol{X}(x_1, x_2, \cdots, x_k)^{\mathrm{T}}$ 以使系统在受到瞬变载荷作用下，在给定时间区间[0, T]内满足瞬时动态性能（振幅或相对位移极限、动应力、动应变破坏极限等）及设计变量允许变化范围等约束条件，并使系统的某些关心位置或坐标的最大动态响应（位移、速度、加速度）在某种意义或准则下达到最优。其数学模型可表示为

$$\left.\begin{aligned}
&\boldsymbol{X}=(x_1,x_1,\cdots,x_k)^{\mathrm{T}}\\
&\boldsymbol{z}(t)=[z(t),\dot{z}(t),\ddot{z}(t)]^{\mathrm{T}}\\
&\dot{z}(t)=p[X,t,z(t),F(t)]\\
&z(t_0)=z_0,\dot{z}(t_0)=\dot{z}_0\\
&J(X,z,t)\\
&h_i[X,t,z(t)]=0\quad i=1,2,\cdots,m\\
&g_j[X,t,z(t)]\leqslant 0\quad j=1,2,\cdots,n\\
&\forall t\in[0,T]
\end{aligned}\right\}\tag{3.32}$$

式中，$\boldsymbol{X}$ 是系统的设计变量向量，由系统的几何参数和物理参数组成；$\boldsymbol{z}(t)$ 是系统的状态变量向量，由系统运动状态的广义坐标，如位移、速度、加速度等组成，它需要满足机械系统运动规律的运动方程，即动态特性的数学描述，$\dot{z}(t)=p[X,t,z(t),F(t)]$；$J$ 是目标函数；h_i、g_j 分别表示等式约束与不等式约束，它们在所有时间点（$\forall t\in[0,T]$）上都要满足。一般我们所建立的运动方程是二阶微分方程或二阶微分方程组，但是这里的数学模型中的运动方程是一阶的，可以用变量代换将二阶微分方程或二阶微分方程组转化为一阶微分方程组。

处理与时间相关的约束有两种方法（见文献[70]）：一种是约束在所有 GLL 点得到满足，即 GLL 点法；另一种是约束在每一个单元的绝对值极值点得到满足，即关键点法。第一种方法要求 GLL 点之间的距离尽量小，这样可以减小漏掉 GLL 之间的绝对值极值点得不到满足的可能性，因此约束数量比较庞大，同时谱元法的求解开销也较大；第二种方法可以用满足精度要求的较少谱单元求解运动微分方程，在每一个单元内，对其高次 Lagrange 函数进行一维搜索找到单元绝对值极值点（当然单元绝对值极值点是振荡的，因此每迭代一步，单元绝对值极值点需要重新计算）。

3.5 实例分析

3.5.1 动态响应分析

为了验证逐步时间谱元法的精度，本书作者选择了几个特定的计算模型，并将谱元法的结果和精确解的结果进行了对比；另外，为了验证逐步时间谱元法的效率，将其求解计算模型的结果与谱元法进行了比较，仿真计算机型号为 TravelMate5530G。

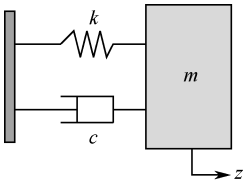


图 3.5　减振器

1. 线性单自由度减振器的动态响应分析

图 3.5 所示的减振器有一个固定质量 m=1kg，弹簧刚度系数和阻尼器系数分别为 k=0.9，c=0.9。当 t=0 时，系统以

速度 v=1m/s 撞击在一个固定的阻碍物上。运动方程为

$$\ddot{z}(t)+c\dot{z}(t)+kz(t)=0 \tag{3.33}$$

式（3.33）的解析解为

$$z(k,c,m,t)=\begin{cases}\dfrac{\mathrm{e}^{-t\frac{c}{2m}}}{\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}}\sin\left(t\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}\right), & 0\leqslant\dfrac{c/(2m)}{\sqrt{k/m}}<1\\ t\mathrm{e}^{-t\sqrt{\frac{k}{2m}}}, & \dfrac{c/(2m)}{\sqrt{k/m}}=1\\ \dfrac{\mathrm{e}^{-t\frac{c}{2m}}}{\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}}\left[\mathrm{e}^{t\sqrt{\left(\frac{c}{2m}\right)^2-\frac{k}{m}}}-\mathrm{e}^{-t\sqrt{\left(\frac{c}{2m}\right)^2-\frac{k}{m}}}\right], & \dfrac{c/(2m)}{\sqrt{k/m}}>1\end{cases} \tag{3.34}$$

设 $x_1(t)=\dot{z}(t),x_2(t)=z(t)$，则式（3.33）可表达为

$$\begin{cases}\dot{x}_1(t)+cx_1(t)+kx_2(t)=0\\ \dot{x}_2(t)-x_1(t)=0\end{cases} \tag{3.35}$$

将式（3.35）转化为矩阵形式：

$$\begin{Bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{Bmatrix}+\begin{bmatrix}c & k\\ -1 & 0\end{bmatrix}\begin{Bmatrix}x_1(t)\\ x_2(t)\end{Bmatrix}=0 \tag{3.36}$$

$$x_1(0)=1,x_2(0)=0$$

减振器的位移响应如图 3.6 所示。

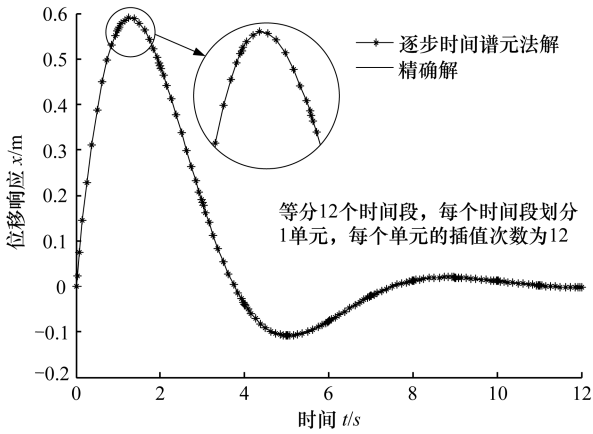


图 3.6　减振器的位移响应

由表 3.1 可知，采用时间谱元法计算动力学问题能够保证足够精度，即使单元

很少，精度也是非常高的，其中整体时间谱元法较逐步时间谱元法有较高的精度，但在一个数量级。整体时间谱元法将每个时间段划分为 12 个单元，插值次数为 12 次；逐步时间谱元法分为 12 个时间段，每一段划分 1 个单元，当每个单元的插值次数为 12 次时，精度为 10^{-15} 数量级。从工程应用角度来说，这个数量级已经足够了。因此，效率成为谱元法最关注的方面。由表 3.1 还可知逐步时间谱元法比整体时间谱元法的效率高。虽然当自由度少、单元少时，逐步时间谱元法的 CPU 时间比整体时间谱元法的 CPU 时间多，但是工程问题的自由度和单元数都非常大。在表 3.1 中，从逐步时间谱元法将每个时间段划分为 10 个单元，整体时间谱元法将每个时间段划分为 120 个单元开始，逐步时间谱元法显示出了其高效性。

表 3.1　线性单自由度减振器的逐步时间谱元法和整体时间谱元法的精度及效率比较

逐步时间谱元法和整体时间谱元法							
单元数		插值数		2 范数精度		CPU 时间（s）	
逐步	整体	逐步	整体	逐步	整体	逐步	整体
1	12	12	12	6.841172e−15	3.473126e−15	1.406250e−01	9.375000e−02
2	24	12	12	5.190670e−15	7.699930e−15	2.343750e−01	1.562500e−01
4	48	12	12	2.135149e−14	1.816118e−14	3.906250e−01	3.281250e−01
4	48	12	12	2.135149e−14	1.816118e−14	4.375000e−01	3.593750e−01
10	120	12	12	1.033398e−13	8.765756e−14	8.437500e−01	1.031250e+000
20	240	12	12	3.352256e−13	2.533890e−13	1.546875e+000	2.828125e+000
30	360	12	12	6.485045e−13	4.937006e−13	2.296875e+000	5.687500e+000
50	600	12	12	1.317409e−12	1.000532e−12	4.125000e+000	1.507813e+001
50	600	10	12	2.719223e−12	1.000532e−12	3.109375e+000	1.512500e+001
50	600	10	10	2.719223e−12	3.523391e−12	3.031250e+000	8.531250e+000
50	600	16	16	5.545065e−12	5.410217e−12	6.984375e+000	4.059375e+001
100	1200	6	6	2.521551e−12	2.868561e−12	2.734375e+000	6.562500e+000
200	2400	6	6	7.034602e−12	8.212221e−12	6.250000e+000	2.443750e+001
200	2400	12	12	1.143609e−11	8.700487e−12	2.556250e+001	2.263438e+002

2. 悬臂梁动态响应分析

悬臂梁的振动有许多应用背景。例如，最近在微型飞行器中的应用[50]，是用机翼的周期共振来驱动飞行的。其中，机翼被看成非线性梁来计算机翼颤动的精确时间响应。主导悬臂梁柔性振动的偏微分方程在空间上是四阶的[52]。在空间上，由传统的有限元法离散微分方程可形成二次微分方程组[53、54]：

$$M\{\ddot{w}\}+K\{w\}=f(t) \tag{3.37}$$

式中，$\dot{w}(0)=0$（为该方程的初始条件），$w(0)=0$；$\{w\}$ 为节点变形。每一个节

点包括 6 个变形：3 个位移 $\{x,y,z\}$ 和 3 个旋转 $\{\theta_x,\theta_y,\theta_z\}$。这里没有考虑阻尼。下面研究悬臂梁的几何尺寸和物理参数，如表 3.2 所示。

表 3.2　悬臂梁的特征和载荷振幅

I_z，cm^4	I_y，cm^4	E，$\mathrm{N/cm}^2$	G, $\mathrm{N/cm}^2$
1450	1296	10^7	3.9×10^6
A，cm^2	L，cm	ρ，$\mathrm{kg/cm}^3$	b，N
19.5	100	2.588×10^{-4}	3.0×10^4

表中，I_z，I_y 为截面惯性矩；E 为弹性模量；G 为剪切模量；A 为梁截面积；L 为梁的长度；ρ 为梁的密度；b 为激励力的振幅。

悬臂梁在空间上用欧拉梁单元来离散，每一个节点有 6 个自由度。这里使用 ANSYS[62]获得 M 和 K。下面把悬臂梁离散为 10 个单元，即 11 个节点，左端固定，因此共有 60 个自由度。然后用谱元法分析计算悬臂梁的瞬态响应。悬臂梁的端点受 $f(t)=b\sin(\omega t)$ 的力，其方向为竖直向上，如图 3.7 所示。

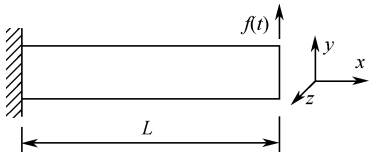


图 3.7　悬臂梁

要想用谱元法解方程式（3.37），首先必须将二次微分方程转化为一次形式，设 $x_1=\dot{w}$，$x_2=w$ 分别为沿着节点速度和节点位移集合，则有

$$\begin{Bmatrix}\dot{x}_1\\ \dot{x}_2\end{Bmatrix}+\begin{bmatrix}\boldsymbol{M} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}0 & \boldsymbol{K}\\ -\boldsymbol{I} & 0\end{bmatrix}\begin{Bmatrix}x_1\\ x_2\end{Bmatrix}=\begin{bmatrix}\boldsymbol{M} & 0\\ 0 & \boldsymbol{I}\end{bmatrix}^{-1}\{f(t)\} \tag{3.38}$$

式中，$\{f(t)\}=\left[\left[\underbrace{0\cdots0}_{n_1}\ b\sin\omega t\ 0\ 0\ 0\ 0\right]\left[\underbrace{0\cdots0}_{n_2}\right]\right]$，$n_1=55$，$n_2=60$。

此悬臂梁的前 9 阶固有频率如表 3.3 所示。

表 3.3　悬臂梁的前 9 阶固有频率（Hz）

第一阶	88.31903	第四阶	491.9311	第七阶	922.7896
第二阶	93.25253	第五阶	509.5645	第八阶	1280.999
第三阶	305.0827	第六阶	533.3691	第九阶	1328.993

悬臂梁的逐步时间谱元法、整体时间谱元法与 ANSYS/Newmark 法的效率比较如表 3.4 所示。

表 3.4　悬臂梁的逐步时间谱元法、整体时间谱元法与 ANSYS/Newmark 法的效率比较

整体时间谱元法和逐步时间谱元法					
单元数		插值数		CPU 时间（s）	
逐步	整体	逐步	整体	逐步	整体
1	10	10	10	3.156250e+000	4.484375e+000
2	20	10	10	6.453125e+000	9.890625e+000
3	30	10	10	9.484375e+000	1.467188e+001
4	40	10	10	1.218750e+001	1.968750e+001
4	40	16	16	2.767188e+001	4.007813e+001
5	50	10	10	1.618750e+001	2.479688e+001
10	100	10	10	3.732813e+001	4.978125e+001
10	100	16	16	7.848438e+001	1.173750e+002
20	200	10	10	7.646875e+001	1.026094e+002
20	200	16	16	1.527969e+002	2.656250e+002
30	300	10	10	1.145781e+002	1.941875e+002
ANSYS/Newmark 算法				时间步长	CPU 时间（s）
				0.001	389
				0.0005	571
				0.0002	1498
				0.0001	2783

因为悬臂梁的一阶固有频率为 88.31903Hz，所以首先计算激励力频率远离固有频率的一个频率为 5Hz（ω=10π）的瞬态响应。如图 3.8 所示，整体时间谱元法

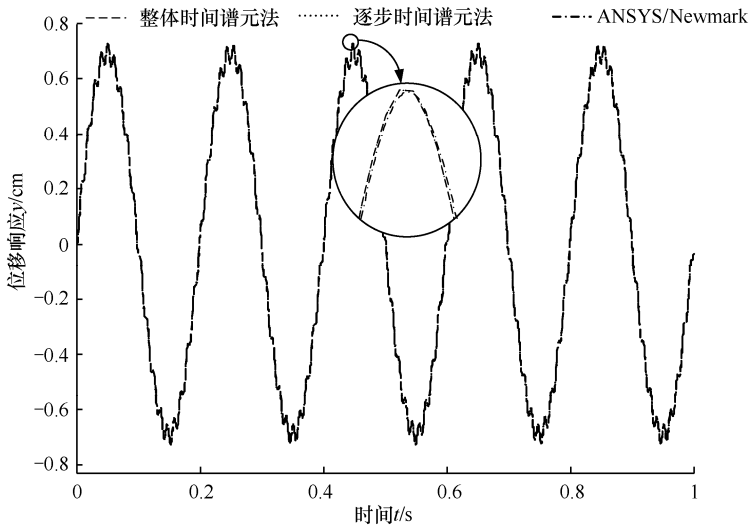


图 3.8　在频率为 5Hz（ω=10π）的简谐激励力作用下，悬臂梁最右端在竖直方向的位移

取 200 个单元，每一个单元进行 10 次插值；逐步时间谱元法划分 10 个时间段，每个时间段划分 20 个单元，每个单元进行 10 次插值；ANSYS/Newmark 法的时间步长为 0.0001s，图中对三者进行了比较。图 3.9 所示是在频率为 88.32Hz（ω=176.64π）的简谐激励力作用下，悬臂梁最右端在竖直方向的位移。

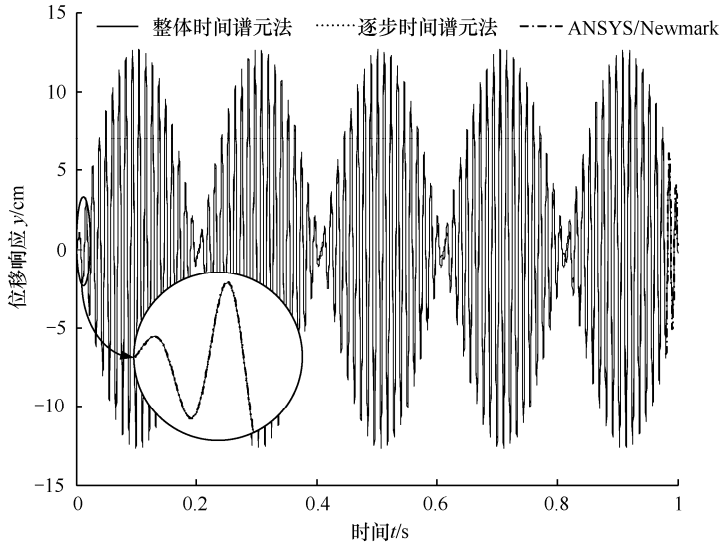


图 3.9　在频率为 88.32Hz（$\omega=176.64\pi$）的简谐激励力作用下，悬臂梁最右端在竖直方向的位移

从图 3.8 可以看出，悬臂梁的无阻尼响应是周期性的，但是其频率包括悬臂梁的一阶固有频率和激励力的频率两部分，当激励力的频率远离一阶固有频率时，悬臂梁的响应比较小，响应近似于以激励力为频率的周期运动。采用谱元法求解瞬态响应的振幅和相位是合理的。

当激励力频率非常接近一阶固有频率时，从图 3.9 可以看出，瞬态响应分析虽然近似于周期响应，但是瞬态分析出现共振现象，振幅非常大，由此也证明了设计结构时要使结构的固有频率避开激励力频率的规律。

由表 3.4 还可知，谱元法求解结构动态响应的效率比 ANSYS/Newmark 法更有效，仿真时间相差一个数量级。而逐步时间谱元法的效率又比整体时间谱元法的效率高。

3.5.2　动态响应优化

在各种各样的机械系统（如减振器、武器后座机构、飞机起落架、汽车悬架系统等）中，系统的主质量在该系统达到稳态之前可能做大幅度振动，当激励力的频率接近于主系统的固有频率时，系统可能发生破坏，因此对激励的瞬时动态

响应必须进行约束。对于这种问题，可以用物理模型的优化来解决。

1. 线性单自由度减振器的最优设计

图 3.10 所示的线性单自由度系统有一个固定质量 m=1kg 和两个设计变量 k 和 c，分别为弹簧刚度系数和阻尼器系数。当 t=0 时，系统以速度 v=1m/s 撞击在一个固定的阻碍物上。设计 k 和 c，使得在时间[0, 12s]内质量块的加速度极小化，满足位移最大响应不大于 1m 的要求。

运动方程为

$$\ddot{z}(t)+c\dot{z}(t)+kz(t)=0 \tag{3.39}$$

式（3.39）的解析解为

$$z(k,c,m,t)=\begin{cases}\dfrac{\mathrm{e}^{-t\frac{c}{2m}}}{\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}}\sin\left(t\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}\right), & 0\leqslant\dfrac{c/(2m)}{\sqrt{k/m}}<1\\ t\mathrm{e}^{-t\sqrt{\frac{k}{2m}}}, & \dfrac{c/(2m)}{\sqrt{k/m}}=1\\ \dfrac{\mathrm{e}^{-t\frac{c}{2m}}}{\sqrt{\dfrac{k}{m}-\left(\dfrac{c}{2m}\right)^2}}\left[\mathrm{e}^{t\sqrt{\left(\frac{c}{2m}\right)^2-\frac{k}{m}}}-\mathrm{e}^{-t\sqrt{\left(\frac{c}{2m}\right)^2-\frac{k}{m}}}\right], & \dfrac{c/(2m)}{\sqrt{k/m}}>1\end{cases} \tag{3.40}$$

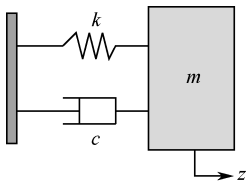


图 3.10　减振器

（1）目标函数的处理

引入人工设计变量 b，则线性单自由度减振器问题的优化模型为

$$\left.\begin{aligned}&\min b\\&\ddot{z}(t)+c\dot{z}(t)+kz(t)=0\\&|c\dot{z}(t)+kz(t)|-b\leqslant 0\\&|z(t)|-1\leqslant 0\end{aligned}\right\} \tag{3.41}$$

（2）处理与时间相关的约束

处理与时间相关的约束有两种方法：一种是约束在所有 GLL 点得到满足，即 GLL 点法；另一种是约束在每一个单元的绝对值极值点得到满足，即关键点法。

第一种方法要求 GLL 点之间的距离尽量小，这样可以减小漏掉 GLL 点之间的绝对值极值点不满足的可能性，因此约束数量比较庞大，同时谱元法的求解开销也较大；第二种方法可以用满足精度要求的较少的谱单元求解运动微分方程，在每一个单元内，对其高次 Lagrange 函数进行一维搜索找到单元绝对值极值点（当然单元绝对值极值点是振荡的，因此每迭代一步，单元绝对值极值点需要重新计算）。

将优化模型式（3.41）转化为包含式（3.16）的优化模型，然后分别应用解析法和谱元法求解。解析解结果如图 3.11 所示。显然，图中的实心圆即为最优点。谱元法求解结果如表 3.5～表 3.8 和图 3.12、图 3.13 所示，其中表 3.5、表 3.6 和图 3.12 是采用 GLL 点法得出的结果，表 3.7、表 3.8 和图 3.13 是采用关键点法得出的结果。表 3.9 是本节结果与精确值的比较。

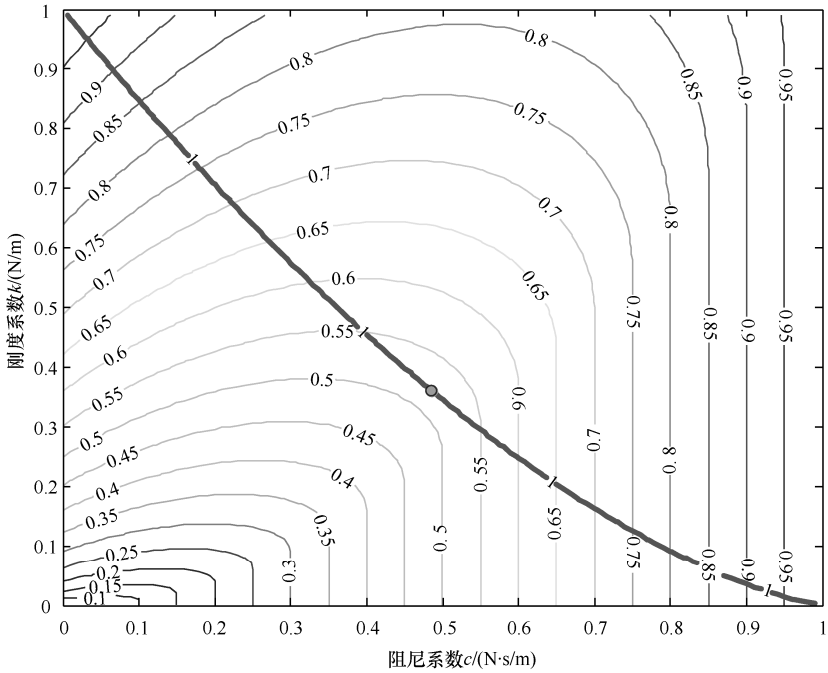


图 3.11　减振器问题的解析解

1）GLL 点法解线性单自由度减振器最优设计问题

表 3.5　GLL 点法的数值试验最优点（单自由度）

			单元数 Nel						
			10	20	60	100	200	300	400
插值		c	0.4684655	0.487330	0.484996	0.484435	0.4840003	0.485851	0.485281
点数	3	k	0.3746571	0.355675	0.360456	0.361225	0.3617721	0.359815	0.360431
m		$\lvert a\rvert$	0.5124303	0.518689	0.520384	0.520524	0.5205923	0.520584	0.520593

（续表）

			单元数 Nel						
			10	20	60	100	200	300	400
插值点数 m	6	c	0.4984423	0.485088	0.484061	0.486063	0.4848087	0.485537	0.485042
		k	0.3464638	0.360561	0.361708	0.359608	0.36093	0.360162	0.360684
		$\|a\|$	0.5195993	0.520487	0.520592	0.520590	0.520596	0.520596	0.520597
	10	c	0.4939818	0.484161	0.485248	0.484568	0.4848257	0.485077	0.485282
		k	0.3510828	0.361601	0.360467	0.361183	0.3609125	0.360646	0.360431
		$\|a\|$	0.5201820	0.520586	0.520592	0.520598	0.5205981	0.520597	0.520598

			插值点数 m		
			3	6	10
单元数 Nel	500	c	0.4849336	0.4855214	0.4849362
		k	0.3607923	0.3601791	0.3607958
		$\|a\|$	0.5205929	0.5205979	0.5205983
	600	c	4.847042e−001	0.4851940	0.4851150
		k	3.610399e−001	0.3605249	0.3606082
		$\|a\|$	5.205972e−001	0.5205980	0.5205984
	800	c	0.4851674	0.4852141	0.4851482
		k	0.3605523	0.3605036	0.3605730
		$\|a\|$	0.5205977	0.5205983	0.5205986

*表中的$|a|$为加速度。

表 3.6　GLL 点法的 CPU 耗时（s）及迭代次数（单自由度）

			单元数 Nel						
			10	20	60	100	200	300	400
插值点数 m	3	t	5.468750	3.328125	7.67187	15.4062	75.8750	47.78125	46.10938
		n	8	10	10	12	20	20	15
	6	t	3.468750	5.421875	24.3750	70.5312	426.656	186.0469	123.5156
		n	10	11	15	20	27	39	18
	10	t	4.218750	14.10938	66.9062	289.421	201.3281	170.7031	633.6250
		n	9	15	18	27	31	15	37

			插值点数 m		
			3	6	10
单元数 Nel	500	t	105.2656	436.4063	1872.672
		n	29	53	69
	600	t	184.7656	221.0781	521.7031
		n	44	21	16
	800	t	86.12500	249.6563	585.389
		n	13	16	105

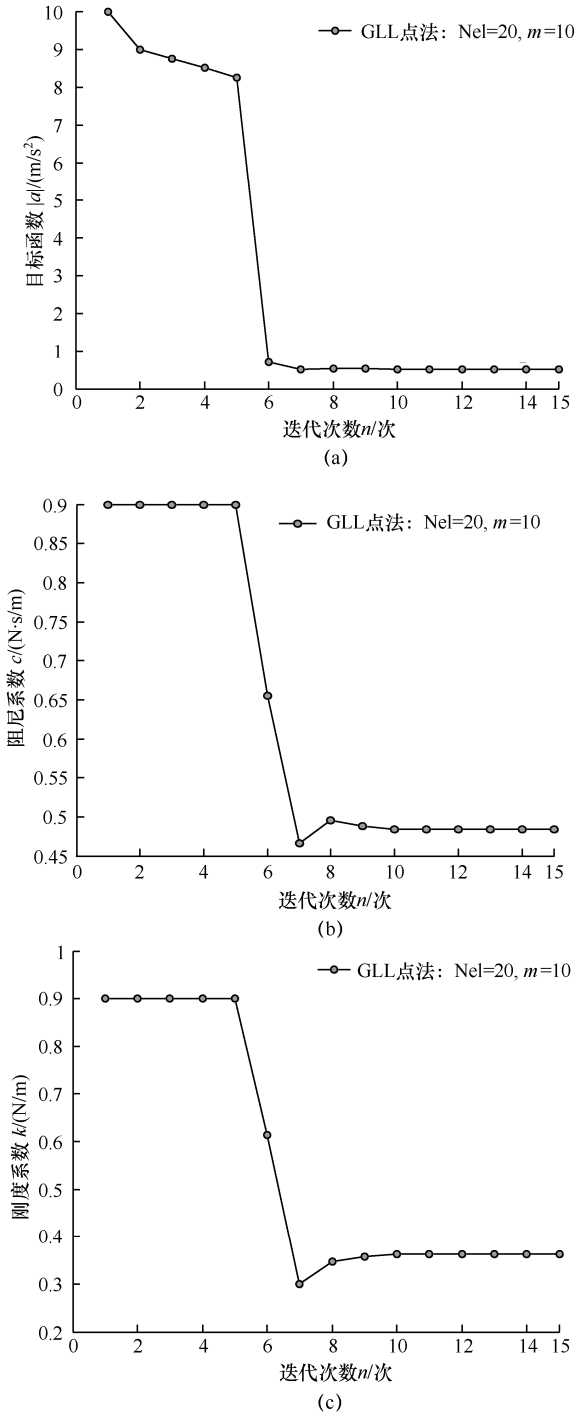


图 3.12　不同单元数和插值次数的优化迭代

应用 GLL 点法处理约束，应用谱元法求解线性单自由度减振器最优设计问题时，从表 3.5 和表 3.6 可以看出，除了单元数 10、20、60，插值数 3 和单元数 10，插值数 3、6、10 没有获得精确最优值之外，其他情况都获得了最优解，其中当单元数大于 200 时，得到的结果最精确。图 3.12 说明了 GLL 点法处理约束的谱元法能快速收敛到最优点。

2）关键点法解线性单自由度减振器最优设计问题

表 3.7　关键点法的数值试验最优点（单自由度）

			单元数 Nel						
			10	20	60	100	200	300	400
插值点数 m	3	c	0.484269	0.484827	0.485112	0.485137	0.485148	0.485149	0.48515
		k	0.359551	0.360387	0.360550	0.360563	0.360568	0.360569	0.36057
		$\|a\|$	0.519002	0.520154	0.520547	0.520580	0.520594	0.520597	0.52059
	6	c	0.485036	0.485122	0.485148	0.485150	0.485151	0.485151	0.48515
		k	0.360512	0.360555	0.360569	0.360570	0.360570	0.360570	0.36057
		$\|a\|$	0.520447	0.520560	0.520594	0.520597	0.520598	0.520598	0.52059
	10	c	0.485134	0.485147	0.485151	0.485151	0.485151	0.485151	0.48515
		k	0.360562	0.360568	0.360570	0.360570	0.360570	0.360570	0.36057
		$\|a\|$	0.520576	0.520593	0.520598	0.520598	0.520599	0.520599	0.52059

			插值点数 m		
			3	6	10
单元数 Nel	500	c	0.485150	0.485151	0.485151
		k	0.360570	0.360570	0.360570
		$\|a\|$	0.520598	0.520599	0.520599
	600	c	0.485151	0.485151	0.485151
		k	0.360570	0.360570	0.360570
		$\|a\|$	0.520598	0.520599	0.520599
	800	c	0.485151	0.485151	0.485151
		k	0.360570	0.360570	0.360570
		$\|a\|$	0.520598	0.520599	0.520599

表 3.8　关键点法的 CPU 耗时（s）及迭代次数（单自由度）

			单元数 Nel						
			10	20	60	100	200	300	400
插值点数 m	3	t	16.843750	27.312500	78.484375	143.140625	265.17187	423.0468	538.8593
		n	12	12	12	13	14	15	15
	6	t	16.968750	34.156250	114.015625	239.015625	296.95312	686.8906	717.00000
		n	10	11	13	16	11	18	13
	10	t	24.578125	56.500000	163.593750	255.000000	517.01562	1064.531	1170.9843
		n	11	13	13	13	13	17	16

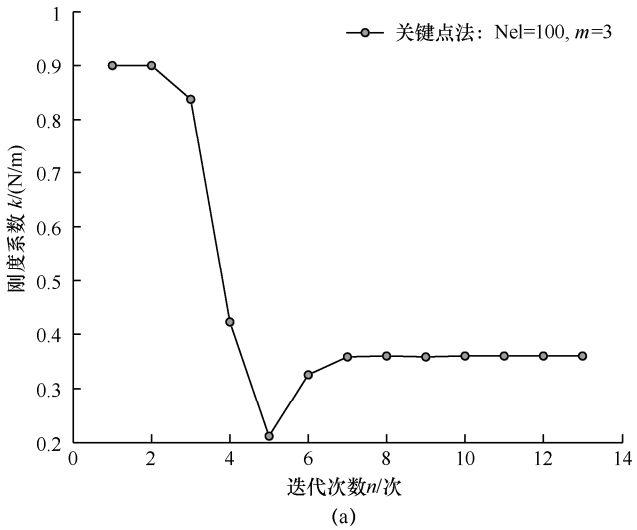
（续表）

			插值点数 m		
			3	6	10
单元数 Nel	500	t	438.984375	691.421875	1760.796875
		n	12	13	15
	600	t	538.046875	853.437500	1495.906250
		n	13	14	14
	800	t	781.421875	1122.609375	2910.078125
		n	14	13	14

表 3.9　本节结果与精确值的比较

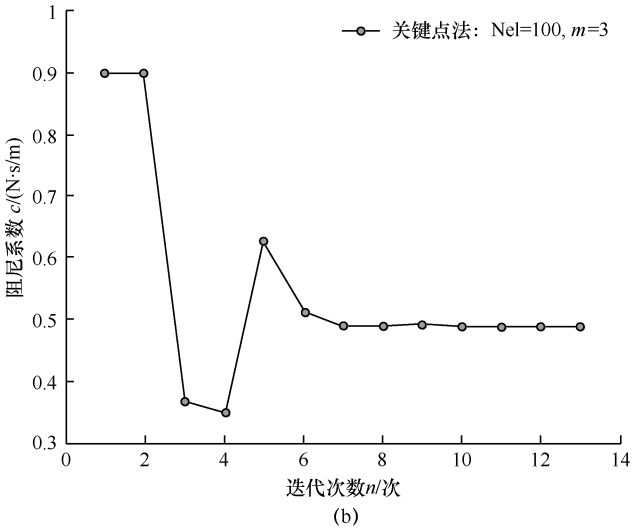
	本　节			解　析　解
	初　始　值	最　优　值		
		GLL 点法	关键点法	
阻尼系数 k	0.9	0.4852141	0.485151	0.4851
刚度系数 c	0.9	0.3605036	0.360570	0.3606
目标函数 d	10	0.5205983	0.520598	0.5206

从表 3.5 和表 3.6 的分析可以看出，在 GLL 点法的实验结果中，除了单元数 10，插值数 3、6 和单元数 20，插值数 3 外，都获得了最优点，且合理结果在表 3.5 中呈现出阶梯形状。在合理结果中，耗时最少的是 4.218750s，最多的是 585.3890s。



(a)

图 3.13　不同单元数和插值次数的优化迭代



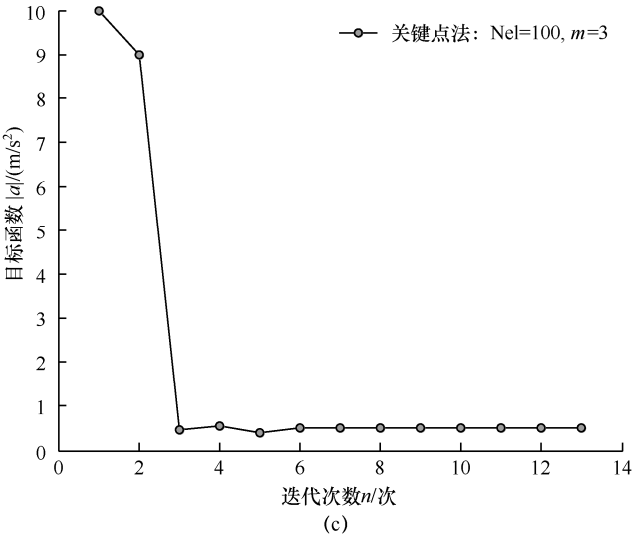


图 3.13　不同单元数和插值次数的优化迭代（续）

从表 3.7 和表 3.8 的分析可以看出，在关键点法的实验结果中，除了单元数 10、插值数 3 外，都获得了最优点。在正确的结果中，耗时最少的是 16.968750s，最多的是 2910.078125s。图 3.13 说明用关键点法处理约束的谱元法也能快速收敛到最优点。从图 3.14 可以发现关键点法处理约束的谱元法开始很快下降到最优点附近，而 GLL 点法处理约束的谱元法比较慢，但是最终 GLL 点法较关键点的迭代步骤少。因此，从准确性来看，关键点法优于 GLL 点法；从耗时来看，GLL 点法优于关键点法。图 3.15 说明优化结果非常明显。

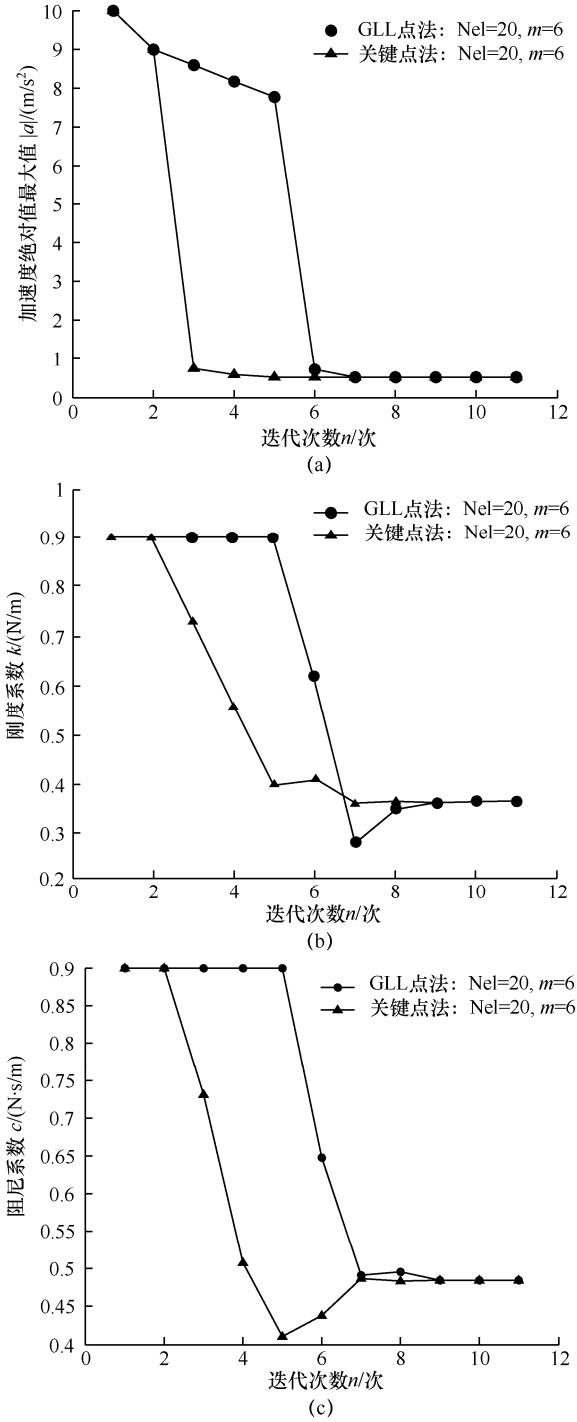


图 3.14　GLL 点法和关键点法的比较

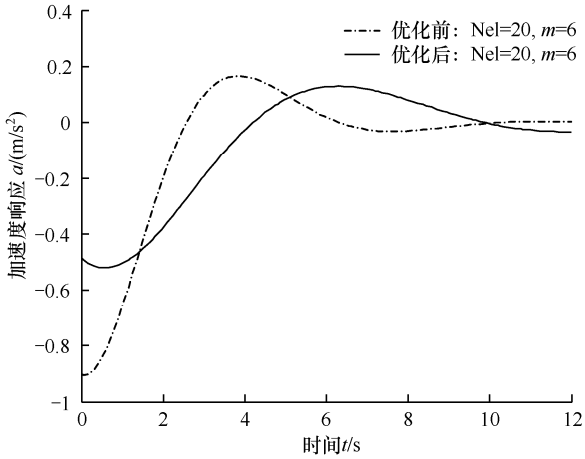


图 3.15　优化前后的加速度

2. 线性两自由度减振器的最优设计

已知两自由度系统的参数如图 3.16 所示，m_1=4.534kg，m_2=0.4534kg，k_1=1749.03N/m，激励频率是主质量固有频率 Ω_n 的 1.2 倍，即 $\omega = 23.57$。求阻尼器的刚度系数 k_2 和阻尼系数 c 使得主质量的最大位移响应最小，同时满足振动空间约束条件（减振器位移响应和主质量位移响应之差不能超过 3 倍的主质量最大响应）及稳态约束条件。初始条件均为零。

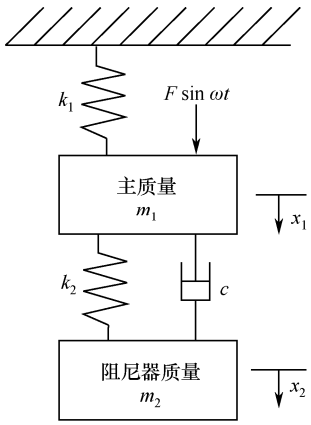


图 3.16　减振器

$$\left.\begin{aligned} &m_1\ddot{x}_1(t)+k_1x_1(t)+k_2[x_1(t)-x_2(t)]+c[\dot{x}_1(t)-\dot{x}_2(t)]=F\sin\omega t\\ &m_2\ddot{x}_2(t)+k_2[x_2(t)-x_1(t)]+c[\dot{x}_2(t)-\dot{x}_1(t)]=0\\ &x_1(0)=x_2(0)=0,\dot{x}_1(0)=\dot{x}_2(0)=0 \end{aligned}\right\} \quad (3.42)$$

引入符号：$x_{st}=\sqrt{F/k_1}$ 表示由力 F 产生的主质量的静位移；$\Omega_n=\sqrt{k_1/m_1}$ 表示

主系统的非耦合固有频率；$\omega_n=\sqrt{k_2/m_2}$ 表示阻尼器系统的非耦合固有频率；$\bar{\mu}=m_2/m_1$ 表示减振器质量与主质量之比；$f=\omega_n/\Omega_n$ 表示减振器与主质量的非耦合固有频率之比；$\zeta=\omega/\Omega_n$ 表示激励频率与主质量的非耦合固有频率之比；$C_c=2m_2\omega_n$ 表示临界阻尼；c 表示阻尼系数；$\xi=c/C_c$ 表示阻尼比；$x_1=x_1(\xi,f,\zeta)$ 表示主质量的位移。令 $z_i=x_i/x_{st}$，i=1,2，则式（3.42）可转化为

$$\left.\begin{aligned}&\ddot{z}_1+\Omega_n^2 z_1+2\bar{\mu}\xi\omega_n(\dot{z}_1-\dot{z}_2)+\bar{\mu}f^2\Omega_n^2(z_1-z_2)=\Omega_n^2\sin(\Omega\zeta\tau)\\&\ddot{z}_2+2\xi\omega_n(\dot{z}_2-\dot{z}_1)+f^2\Omega_n^2(z_2-z_1)=0\end{aligned}\right\}\tag{3.43}$$

再令 $x_1=\dot{z}_1$，$x_2=z_1$，$x_3=\dot{z}_2$，$x_4=z_2$，则式（3.43）可转化为

$$\dot{X}+A_sX=F\tag{3.44}$$

式中，$\dot{X}=\begin{bmatrix}\dot{x}_1 & \dot{x}_2 & \dot{x}_3 & \dot{x}_4\end{bmatrix}^{\mathrm{T}}$，$X=\begin{bmatrix}x_1 & x_2 & x_3 & x_4\end{bmatrix}^{\mathrm{T}}$，

$$\boldsymbol{A}_s=\begin{bmatrix}2\bar{\mu}\xi\omega_n & \Omega_n^2(1+\bar{\mu}f^2) & -2\bar{\mu}\xi\omega_n & -\bar{\mu}f^2\Omega_n^2\\-1 & 0 & 0 & 0\\-2\xi\omega_n & -f^2\Omega_n^2 & 2\xi\omega_n & f^2\Omega_n^2\\0 & 0 & -1 & 0\end{bmatrix},\quad \boldsymbol{F}=\begin{Bmatrix}\Omega_n^2\sin(\Omega\zeta\tau)\\0\\0\\0\end{Bmatrix}$$

因此，优化模型表示为

$$\left.\begin{aligned}&\min\max_{t\in[0,T]}\left|x_2(\xi,f,t)\right|\\&\text{s.t. }\dot{X}+A_sX=F\\&\left|x_4(\xi,f,t)-x_2(\xi,f,t)\right|\leqslant 3\max\left|x_2(\xi,f,t)\right|\\&x_{st4}(\xi,f,t)\leqslant a\\&\left|x_{st4}(\xi,f,t)-x_{st2}(\xi,f,t)\right|\leqslant 3a\\&\xi_{\min}\leqslant\xi\leqslant\xi_{\max}\\&f_{\min}\leqslant f\leqslant f_{\max}\end{aligned}\right\}\tag{3.45}$$

根据文献[63]，引入人工设计变量 x_5，可以把式（3.45）转化为

$$\left.\begin{aligned}&\bar{\varphi}_0(X)=x_5\\&\text{s.t. }\dot{X}+A_sX=F\\&\left|x_2\right|\leqslant x_5\\&\left|x_4(\xi,f,t)-x_2(\xi,f,t)\right|\leqslant 3x_5\\&x_{st4}(\xi,f,t)\leqslant a\\&\left|x_{st4}(\xi,f,t)-x_{st2}(\xi,f,t)\right|\leqslant 3a\\&\xi_l\leqslant\xi\leqslant\xi_u\\&f_l\leqslant f\leqslant f_u\end{aligned}\right\}\tag{3.46}$$

式中，a=0.03048m；x_{st2}，x_{st4} 分别表示主质量和阻尼器的稳态位移响应，表达式为

$$\left.\begin{aligned} x_{st2} &= \sqrt{\frac{\left(2\zeta\xi\right)^2+\left(\zeta^2-f^2\right)^2}{x_b}} \\ x_{st4} &= \sqrt{\frac{\left(2\zeta\xi\right)^2+f^4}{x_b}} \end{aligned}\right\} \tag{3.47}$$

式中，$x_b=(2\zeta\xi)^2(\zeta^2+\bar{\mu}\zeta^2-1)^2+\left[\bar{\mu}f^2\zeta^2-(\zeta^2-1)(\zeta^2-f^2)\right]^2$。

阻尼比 $\xi\in[10^{-6},0.16785]$，减振器与主质量的非耦合固有频率之比 $f\in[10^{-4},2.0]$，减振器质量与主质量之比 $\bar{\mu}=0.1$ 不变。在关键点法中，为了找到关键点，应用黄金分割法和抛物线法进行一维搜索，将目标函数、时间设计变量的收敛上限设置为 1.0×10^{-15}；为了不漏掉一切可能的关键点，在每一个单元进行两次一维搜索，则要求找到的所有单元关键点满足约束条件。以其中一组数据为例，迭代过程如图 3.13 和图 3.14 所示。由文献[63]可知，梯度投影法计算的已知条件 $\bar{\mu}=0.1$，$a=1.2$，$10^{-4}\leqslant f\leqslant2.0$，$10^{-0.6}\leqslant\xi\leqslant0.16785$，$\zeta$=1.2，初始条件 $f=1.6$，$\xi=0.02$，$d=0.081$，最优结果 $f=1.338$，ξ=0.02121，$d=0.0601$。

GLL 点法的数值试验最优点如表 3.10 所示，GLL 点法的数值试验耗费时间如表 3.11 所示。

表 3.10　GLL 点法的数值试验最优点

			单元数 Nel					
			10	20	30	40	50	60
插值点数 m	1	f	1.4352	1.5072	1.5013	1.2139	1.2139	1.2139
		ξ	0.1679	1e−006	0.0544	0.1579	0.1579	0.15787
		$\lvert x_1\rvert$	0.0280	0.0381	0.2263	0.1639	0.1062	0.095949
	2	f	1.5072	1.3304	1.3471	1.3275	1.3371	1.3559
		ξ	1e−006	1e−006	0.0022	0.0316	1e−006	1e−006
		$\lvert x_1\rvert$	0.3533	0.0570	0.0558	0.0596	0.059658	0.058294
	3	f	1.3194	1.3301	1.3287	1.3288	1.3291	1.3563
		ξ	0.0608	0.0098	0.0243	0.0231	0.021252	1e−006
		$\lvert x_1\rvert$	0.0632	0.0618	0.0602	0.0597	0.059749	0.058701
	4	f	1.3176	1.3507	1.3560	1.3541	1.3563	1.329
		ξ	0.0653	1e−006	1e−006	1e−006	1e−006	0.022194
		$\lvert x_1\rvert$	0.0468	0.0578	0.0586	0.0585	0.058667	0.059738
	5	f	1.3611	1.3586	1.3286	1.3288	1.3558	1.356
		ξ	1e−006	0.0009	0.0247	0.0230	1e−006	1e−006
		$\lvert x_1\rvert$	0.0530	0.0581	0.0597	0.0599	0.05867	0.058709

（续表）

			单元数 Nel					
			10	20	30	40	50	60
插值点数 m	6	f	1.3244	1.3286	1.3573	1.3555	1.3557	1.3563
		ξ	0.0454	0.0247	1e−006	1e−006	1e−006	1e−006
		$\lvert x_1\rvert$	0.0583	0.0596	0.0585	0.0586	0.058686	0.058677

表 3.11　GLL 点法的数值试验耗费时间

		单元数 Nel 及 CPU 耗时（s）					
		10	20	30	40	50	60
插值点数 m	1	0.5156	0.5313	0.5938	0.9219	0.8125	1.1094
	2	0.5	0.7031	6.5781	2.7813	3.625	12.031
	3	0.8594	1.4375	3.125	7.9375	10.344	14.469
	4	0.7344	3.25	5.3438	9.3125	16.859	32.656
	5	0.7188	5.1719	12.6719	16.3281	37.469	34.922
	6	1.4844	6.6406	12.0313	21.3906	34.859	42.734

比较后面提到的表 3.12 与文献[63]的优化结果，我们发现 Nel=6，interp=14 是最好的解，耗时为 185.03s。对于 GLL 点法，当所有 GLL 点要求满足约束条件时，对于小单元的求解是不可能找到可行解的，而采用大单元、小插值却可以找到。从表 3.10 中可以发现，Nel=50，interp=3 是最好的解，而且更接近文献[63]的最优数据，耗时仅仅 10.344s。这说明对于这个两自由度减振器的设计问题，GLL 点法优于关键点法；但是在文献[70]中，求解一个自由度的减振器设计问题时得出关键点法优于 GLL 点法。大家都知道，两自由度问题是最简单的多自由度问题，这样就能说明本书的研究结果更具有实际意义。

其实，从耗时来分析，还可以给 SQP 算法提供解析梯度函数或简单的数值梯度函数，这样函数的评估次数能明显减少[64]；SQP 算法采用差分法计算梯度，每次迭代需要多次进行仿真函数评估（评估次数等于设计变量维数的两倍乘以实例数+1），优化求解很费时[65]；对于基于谱元法的动态响应优化，每评估一次函数值，就要运用谱元法更新一次，很明显单元数越多、插值次数越高，耗时越多，若减少函数评估次数，耗时自然就会下降。对于表 3.10、表 3.12 中的优化结果，有一部分的第二个值是 1e−006，和初始值一样。从文献[70]可知，本书作者研究的单自由度减振器优化结果波动的原因是是否有网格点在响应的绝对值最大点处，文献[70]第 10 页的图 3.4 说明了此问题。对于本章节研究的两自由度减振器来说，从表 3.10 中取 m=3，Nel=50，Nel=60，以优化初始值研究其 GLL 点是否在最值点，得出如图 3.17 和图 3.18 所示的结果。从图 3.17 和图 3.18 中，能看出距离最值点越近，收敛精度越高，相反，距离最值点越远，收敛精度越低。表 3.14 是参考文献中的数据和本节数据的比较，可说明本节所介绍方法的正确性和可行性。

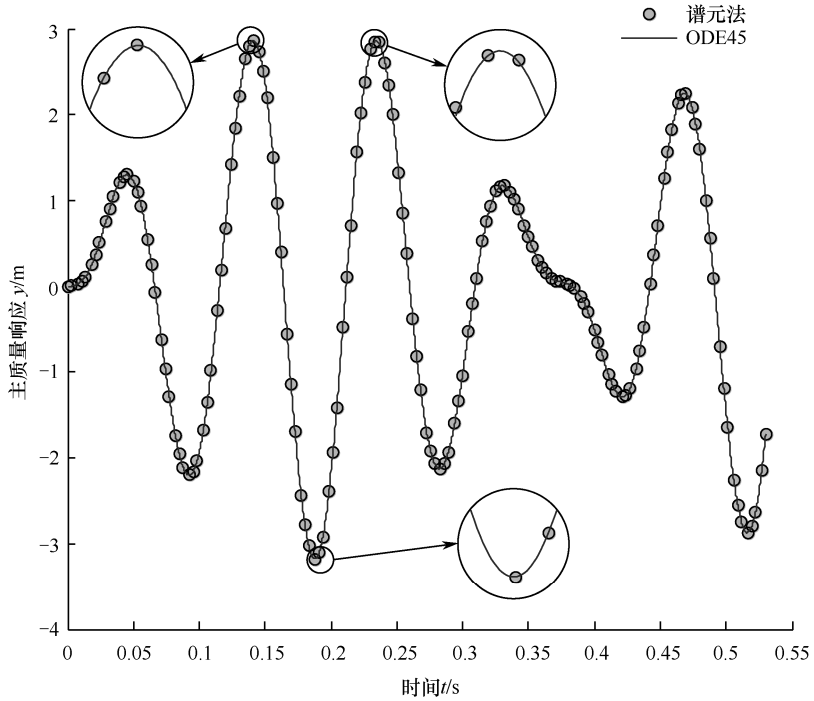


图 3.17　单元数为 50，插值数为 3

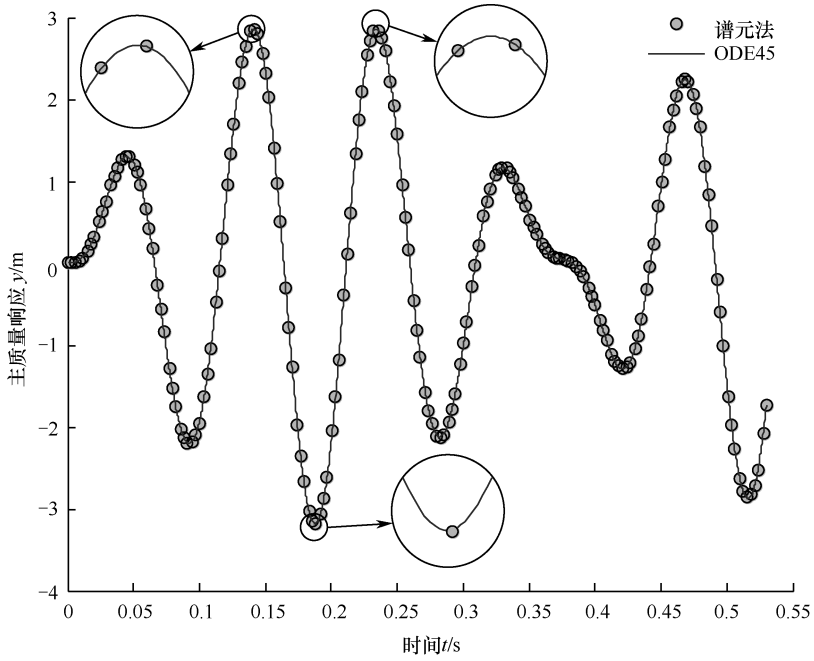


图 3.18　单元数为 60，插值数为 3

固有频率比如图 3.19 所示，阻尼比如图 3.20 所示，主质量的动态响应如图 3.21 所示。

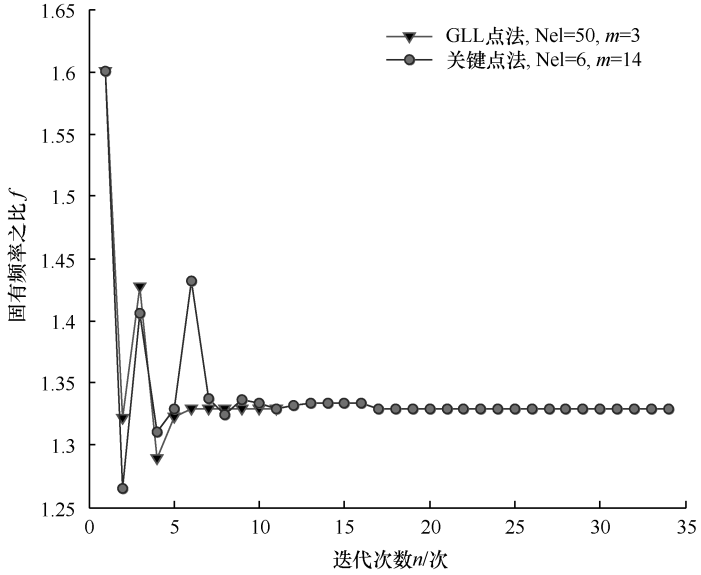


图 3.19　固有频率比

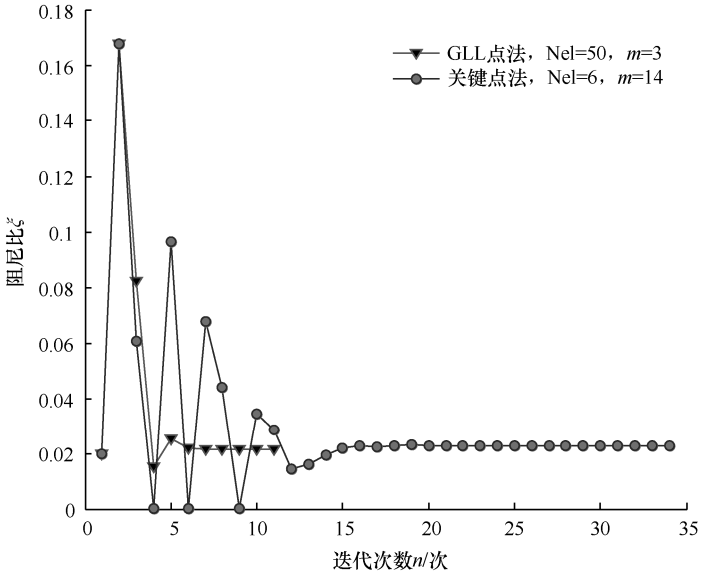


图 3.20　阻尼比

关键点法的数值试验最优点如表 3.12 所示，关键点法的耗时如表 3.13 所示，数据的比较如表 3.14 所示。

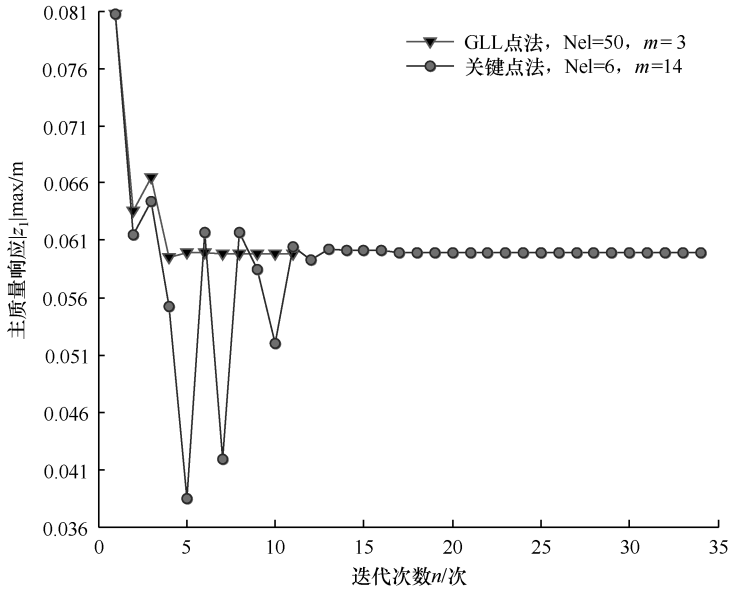


图 3.21　主质量的动态响应

表 3.12　关键点法的数值试验最优点

			单元数 Nel				
			8	10	12	14	16
插值点数 m	1	f	1.2239	1.3304	1.4352	1.3992	2.6235
		ξ	0.1656	1e−006	0.1679	1e−006	0.7913
		$\|x_1\|$	0.026546	0.044478	0.046129	0.035547	0.396984
	2	f	1.2813	1.2270	1.3256	1.3290	1.3801
		ξ	0.1679	0.1679	0.0406	0.0218	0.0088
		$\|x_1\|$	0.068892	0.11877	0.074	0.055093	0.057915
	3	f	1.2547	1.3301	1.3625	1.3261	1.3289
		ξ	0.1400	0.0095	0.0511	0.0384	0.0389
		$\|x_1\|$	0.062794	0.053914	0.055184	0.055062	0.055123
	4	f	1.3770	1.3283	1.3287	1.3274	1.3287
		ξ	1e−006	0.0266	0.0239	0.0319	0.0240
		$\|x_1\|$	0.059766	0.055817	0.056708	0.055674	0.056627
	5	f	1.3708	1.3567	1.3560	1.3289	1.3288
		ξ	1e−006	1e−006	1e−006	0.0229	0.0229
		$\|x_1\|$	0.059492	0.058618	0.058715	0.059883	0.059881
	6	f	1.3288	1.3289	1.3289	1.3289	1.3289
		ξ	0.0233	0.0229	0.0229	0.0229	0.0229
		$\|x_1\|$	0.059827	0.059878	0.059883	0.059883	0.059883

表 3.13　关键点法的耗时（s）

		插值点数 m				
		8	10	12	14	16
单元数 Nel	1	31.95	7.20	5.36	32.05	5.42
	2	70.44	8.41	38.16	63.95	55.58
	3	103.17	56.23	114.58	117.45	87.10
	4	25.27	108.36	135.66	166.70	109.50
	5	157.14	180.91	159.78	202.27	114.60
	6	84.08	115.17	165.92	185.03	230.50

表 3.14　数据的比较

	文献优化结果数据					
	本节	文献[63]	文献[66]	文献[67]	文献[68]	文献[69]
固有频率比 f	1.3289	1.338	1.3277	1.3277	1.3312	1.3597
阻尼系数比 ξ	0.0229	0.02121	0.03054	0.03058	0.02758	0.0184
最大位移响应$\lvert x_1\rvert$	0.059883	0.060122	0.059858	0.059847	0.059842	0.060335

GLL 点法与关键点法的比较如表 3.15～表 3.18 所示。其中，GLL 点法的数值试验最优点如表 3.15 所示。GLL 点法的 CPU 耗时（s）和迭代次数如表 3.16 所示。关键点法的数值试验最优点如表 3.17 所示，关键点法的 CPU 耗时（s）和迭代次数如表 3.18 所示。固有频率比如图 3.22 所示，阻尼比如图 3.23 所示，主质量的动态响应如图 3.24 所示。

表 3.15　GLL 点法的数值试验最优点

			单元数 Nel					
			100	200	300	400	600	800
插值点数 m	3	f	1.356068	1.356083	1.356037	1.356046	1.328857	1.328857
		ξ	0.000001	0.000001	0.000001	0.000001	0.022917	0.022918
		$\lvert x_1\rvert$	0.058681	0.058711	0.058712	0.058709	0.059881	0.059881
	6	f	1.355815	1.356080	1.328860	1.328858	1.328858	1.328857
		ξ	0.000001	0.000001	0.022894	0.022911	0.022908	0.022912
		$\lvert x_1\rvert$	0.058702	0.058712	0.059880	0.059882	0.059882	0.059882
	10	f	1.356054	1.328860	1.328858	1.328857	1.328857	1.328507
		ξ	0.000001	0.022893	0.022906	0.022914	0.022912	0.025423
		$\lvert x_1\rvert$	0.058713	0.059880	0.059881	0.059882	0.059882	0.060091

表 3.16　GLL 点法的 CPU 耗时（s）和迭代次数

			单元数 Nel					
			100	200	300	400	600	800
插值点数 m	3	t	5.765625	4.125000	5.109375	6.171875	31.750000	54.265625
		n	9	12	10	10	38	52
	6	t	4.578125	68.937500	21.359375	49.890625	24.125000	36.796875
		n	11	101	19	38	11	12
	10	t	8.437500	29.828125	22.718750	33.437500	370.375000	610.937500
		n	11	22	11	12	106	124

表 3.17　关键点法的数值试验最优点

			单元数 Nel					
			100	200	300	400	600	800
插值点数 m	3	f	1.328858	1.328857	1.328857	1.328857	1.328857	1.328857
		ξ	0.022909	0.022912	0.022912	0.022912	0.022912	0.022912
		$\lvert x_1\rvert$	0.059884	0.059882	0.059882	0.059882	0.059882	0.059882
	6	f	1.328857	1.328857	1.328857	1.328857	1.328857	1.328857
		ξ	0.022912	0.022912	0.022912	0.022912	0.022912	0.022912
		$\lvert x_1\rvert$	0.059882	0.059882	0.059882	0.059882	0.059882	0.059882
	10	f	1.328857	1.328857	1.328857	1.328857	1.328857	1.328857
		ξ	0.022912	0.022912	0.022912	0.022912	0.022912	0.022912
		$\lvert x_1\rvert$	0.059882	0.059882	0.059882	0.059882	0.059882	0.059882

表 3.18　关键点法的 CPU 耗时（s）和迭代次数

			单元数 Nel					
			100	200	300	400	600	800
插值点数 m	3	t	349.90625	558.20312	807.078125	1263.6562	1000.71875	1163.71875
		n	32	37	35	37	25	24
	6	t	256.32812	444.17187	591.406250	754.43750	1140.82812	2171.03125
		n	26	30	28	28	27	31
	10	t	404.4062	567.3750	656.68750	782.125000	1199.00000	2061.46875
		n	37	31	26	26	27	27

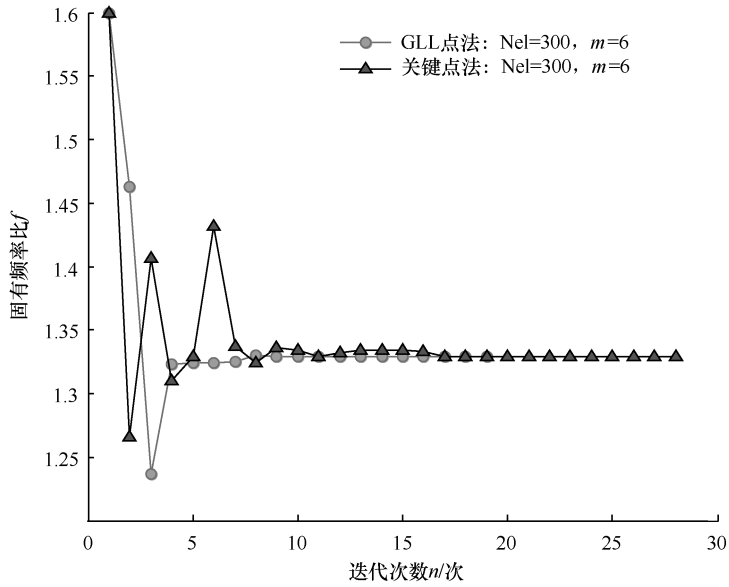


图 3.22　固有频率比

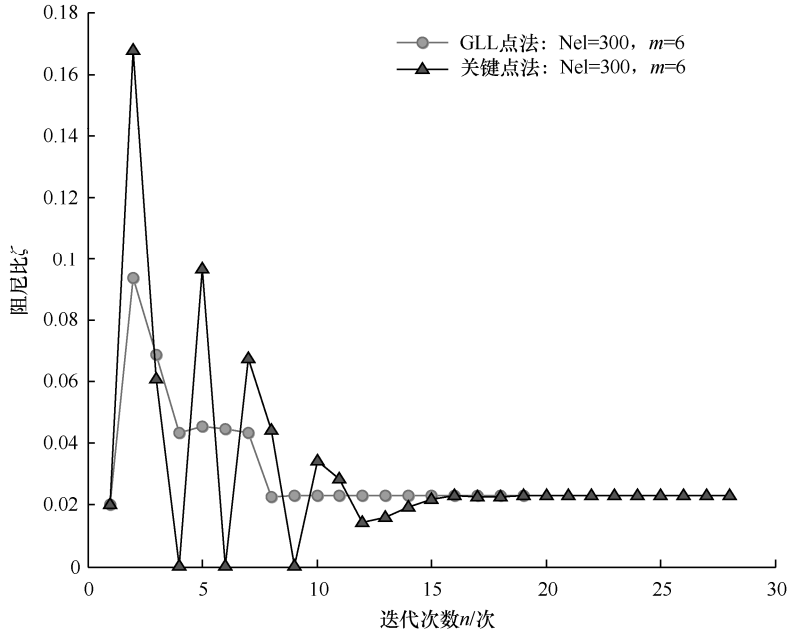


图 3.23　阻尼比

从表 3.15～表 3.18 的分析可以看出，GLL 点法在速度方面有极大的优势，然而在准确度方面呈现阶梯形状，其中在右下方都找到了准确的最优点，而在左上

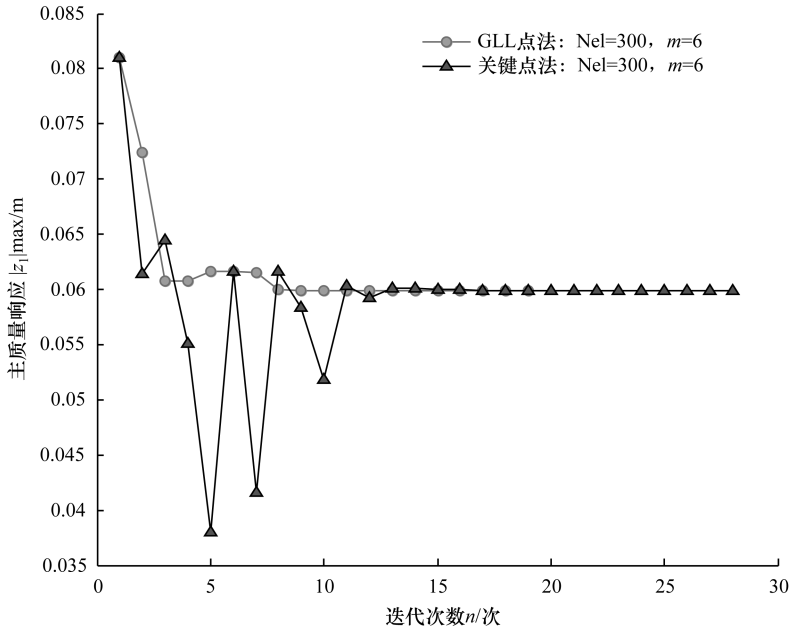


图 3.24　主质量的动态响应

方的阻尼比都非常小，这在设计上是不合理的。在 GLL 点法的实验结果中，单元数为 100、200、300、400，插值点数为 3 的所有结果都是不合理的，单元数为 100、200，插值点数为 6 的试验结果也是不合理的，还有单元数为 100，插值点数为 10 的实验结果也不合理，其余的实验结果都合理（参见表 3.16）。在关键点法的实验结果中，所有的都是合理的（参见表 3.17）。在耗时方面，对 GLL 点法来说，找到最优点花费的最少时间为 21.359375s，单元数为 300，插值点数为 6；最大耗时为 370.375s，对应的单元数为 600，插值点数为 10（参见表 3.16）。对关键点法来说，耗时最少为 256.32812s，单元数为 100，插值点数为 6；最大耗时为 2171.03125s，对应的单元数为 800，插值点数为 6（参见表 3.18）。从图 3.22～图 3.24 来看，GLL 点法优于关键点法，能迅速稳定地找到最优点；但是从准确性来看，关键点法优于 GLL 点法。

3. 汽车悬架系统动态响应优化设计

汽车悬架系统可近似简化为图 3.25 所示的 5 自由度的动力学模型。图 3.25 中的 m_1 是驾驶员及其座位的质量，它由弹簧 k_1 和阻尼器 c_1 支撑在车体上。汽车车体的质量、前后车轴和车轮的质量分别为 m_2、m_4 和 m_5。汽车车体由连于前后车轴的弹簧 k_2 和 k_3 及阻尼器 c_2 和 c_3 支撑。k_4、k_5 和 c_4、c_5 表示轮胎的刚度和阻尼系数。车体对其质量中心的转动惯量用 I 表示。L 表示前后轮距长度。$f_1(t)$和 $f_2(t)$

表示由于汽车行驶道路表面起伏不平所激起的前后轮的位移函数。$y_i(t)$是 5 个自由度的广义坐标。

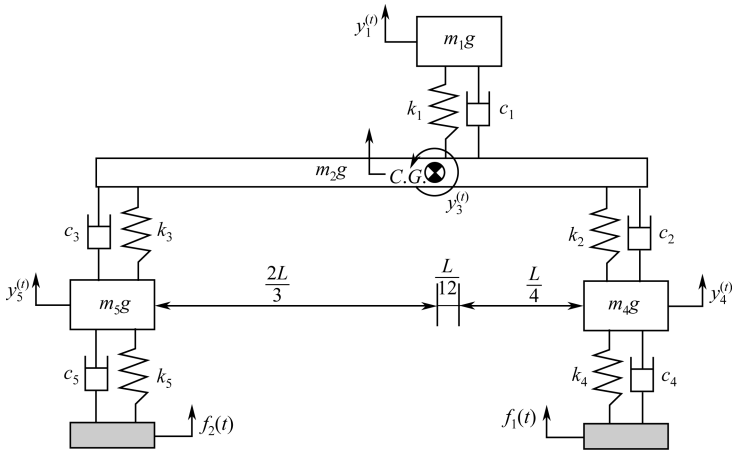


图 3.25　5 自由度的动力学模型

（1）选择设计变量

本优化问题仅取模型中与悬架系统有关的弹簧系数 k_i 和阻尼系统 $c_i(i=1,2,3)$ 作为设计变量，即 $X=(k_1,k_2,k_3,c_1,c_2,c_3)^{\mathrm{T}}$。

（2）建立目标函数

以司机和乘客座位的瞬变动态最小为目标，对其机械系统的结构进行动态响应优化设计，即在设计汽车悬架系统的结构时，要求汽车在各种车速和路面条件下，使得司机和乘客座位的最大加速度响应最小。目标函数可表示为 $f(X,t)=\min[\max|\ddot{y}_1(t)|]$。

（3）满足给定的约束条件

① 满足系统状态方程的约束。

由拉格朗日法建立系统运动的微分方程，再变换为系统的状态方程，即

$$\boldsymbol{M}\ddot{\boldsymbol{y}}+\boldsymbol{C}\dot{\boldsymbol{y}}+\boldsymbol{K}\boldsymbol{y}=\boldsymbol{b}F(t) \tag{3.48}$$

式中，

$$\boldsymbol{M}=\begin{bmatrix} m_1 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & m_4 & 0 \\ 0 & 0 & 0 & 0 & m_5 \end{bmatrix},\quad \boldsymbol{b}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & k_4 & c_4 & 0 & 0 \\ 0 & 0 & 0 & k_5 & c_5 \end{bmatrix}$$

$$\boldsymbol{K}=\begin{bmatrix} k_1 & -k_1 & -\frac{Lk_1}{12} & 0 & 0 \\ -k_1 & k_1+k_2+k_3 & \frac{L}{3}\left(\frac{1}{4}k_1+k_2-2k_3\right) & -k_2 & -k_3 \\ -\frac{Lk_1}{12} & \frac{L}{3}(\frac{1}{4}k_1+k_2-2k_3) & \frac{2L}{9}\left(\frac{1}{16}k_1+k_2+4k_3\right) & -\frac{2Lk_2}{3} & \frac{2Lk_3}{3} \\ 0 & -k_2 & -\frac{Lk_2}{3} & k_2+k_4 & 0 \\ c_1 & -k_3 & \frac{Lk_3}{3} & 0 & k_3+k_5 \end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix} c_1 & -c_1 & -\frac{Lc_1}{12} & 0 & 0 \\ -c_1 & c_1+c_1+c_3 & \frac{L}{3}\left(\frac{1}{4}c_1+c_2-2c_3\right) & -c_2 & -c_3 \\ -\frac{Lc_1}{12} & \frac{L}{3}\left(\frac{1}{4}c_1+c_2-2c_3\right) & \frac{2L}{9}\left(\frac{1}{16}c_1+c_2+4c_3\right) & -\frac{Lc_2}{3} & \frac{2Lc_3}{3} \\ 0 & -c_2 & -\frac{Lc_2}{3} & c_2+c_4 & 0 \\ 0 & -c_3 & \frac{Lc_3}{3} & 0 & c_3+c_5 \end{bmatrix}$$

$$F(t)=\{0 \quad f_1(t) \quad \dot{f}_1(t) \quad f_2(t) \quad \dot{f}_2(t)\}^{\mathrm{T}},\quad y=(y_1 \quad y_2 \quad y_3 \quad y_4 \quad y_5)^{\mathrm{T}}$$

$$\dot{y}=(\dot{y}_1 \quad \dot{y}_2 \quad \dot{y}_3 \quad \dot{y}_4 \quad \dot{y}_5)^{\mathrm{T}},\ddot{y}=(\ddot{y}_1 \quad \ddot{y}_2 \quad \ddot{y}_3 \quad \ddot{y}_4 \quad \ddot{y}_5)^{\mathrm{T}}$$

化简为

$$\ddot{y}+\boldsymbol{C}\dot{y}+\boldsymbol{K}y=\boldsymbol{b}F(t) \tag{3.49}$$

式中，

$$\boldsymbol{b}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{k_4}{m_4} & \frac{c_4}{m_4} & 0 & 0 \\ 0 & 0 & 0 & \frac{k_5}{m_5} & \frac{c_5}{m_5} \end{bmatrix}$$

$$\boldsymbol{K}=\begin{bmatrix}
\frac{k_1}{m_1} & \frac{-k_1}{m_1} & -\frac{Lk_1}{12m_1} & 0 & 0 \\
\frac{-k_1}{m_2} & \frac{k_1+k_2+k_3}{m_2} & \frac{L}{3m_2}\left(\frac{1}{4}k_1+k_2-2k_3\right) & \frac{-k_2}{m_2} & \frac{-k_3}{m_2} \\
-\frac{Lk_1}{12m_3} & \frac{L}{3m_3}\left(\frac{1}{4}k_1+k_2-2k_3\right) & \frac{2L}{9m_3}\left(\frac{1}{16}k_1+k_2+4k_3\right) & -\frac{2Lk_2}{3m_3} & \frac{2Lk_3}{3m_3} \\
0 & \frac{-k_2}{m_3} & -\frac{Lk_2}{3m_3} & \frac{k_2+k_4}{m_4} & 0 \\
c_1 & \frac{-k_3}{m_3} & \frac{Lk_3}{3m_3} & 0 & \frac{k_3+k_5}{m_5}
\end{bmatrix}$$

$$\boldsymbol{C}=\begin{bmatrix}
\frac{c_1}{m_1} & \frac{-c_1}{m_1} & -\frac{Lc_1}{12m_1} & 0 & 0 \\
\frac{-c_1}{m_2} & \frac{c_1+c_2+c_3}{m_2} & \frac{L}{3m_2}\left(\frac{1}{4}c_1+c_2-2c_3\right) & \frac{-c_2}{m_2} & \frac{-c_3}{m_2} \\
-\frac{Lc_1}{12m_3} & \frac{L}{3m_3}\left(\frac{1}{4}c_1+c_2-2c_3\right) & \frac{2L}{9m_3}\left(\frac{1}{16}c_1+c_2+4c_3\right) & -\frac{2Lc_2}{3m_3} & \frac{2Lc_3}{3m_3} \\
0 & \frac{-c_2}{m_4} & -\frac{Lc_2}{3m_4} & \frac{c_2+c_4}{m_4} & 0 \\
c_1 & \frac{-c_3}{m_5} & \frac{Lc_3}{3m_5} & 0 & \frac{c_3+c_5}{m_5}
\end{bmatrix}$$

令 $x=y$，$z=\dot{y}$，则式（3.48）可化简为一阶微分方程组：

$$\left.\begin{aligned}\dot{x}-z&=0\\ \dot{z}+\boldsymbol{C}z+\boldsymbol{K}x&=\boldsymbol{b}F(t)\end{aligned}\right\} \tag{3.50}$$

写成矩阵形式为

$$\begin{Bmatrix}\dot{x}\\ \dot{y}\end{Bmatrix}+\begin{bmatrix}0 & -\boldsymbol{I}_{5\times5}\\ \boldsymbol{K} & \boldsymbol{C}\end{bmatrix}\begin{Bmatrix}x\\ z\end{Bmatrix}=\begin{Bmatrix}0\\ \boldsymbol{b}F(t)\end{Bmatrix} \tag{3.51}$$

② 满足函数形式的约束。

为了提高司机（乘客）的座位舒适性，要求其最大绝对加速度最小，即

$$\ddot{y}_1\leqslant d \tag{3.52}$$

式中，d 是人工变量。

要求限制其车体振幅在一定范围内，即：

车体与司机座位间的相对位移约束为

$$\left|y_2(t)+\frac{L}{12}y_3(t)-y_1(t)\right|\leqslant\theta_1 \tag{3.53}$$

车体与前轮之间的相对位移约束为

$$\left|y_4(t)-y_2(t)-\frac{L}{3}y_3(t)\right|\leqslant\theta_2 \tag{3.54}$$

车体与后轮之间的相对位移约束为

$$\left|y_5(t)-y_2(t)-\frac{2L}{3}y_3(t)\right|\leqslant\theta_3 \tag{3.55}$$

路面与前轮之间的相对位移约束为

$$\left|y_4(t)-f_1(t)\right|\leqslant\theta_4 \tag{3.56}$$

路面与后轮之间的相对位移约束为

$$\left|y_5(t)-f_2(t)\right|\leqslant\theta_5 \tag{3.57}$$

式中，$\theta_1\sim\theta_5$ 为最大允许相对位移。

③ 满足设计变量变化范围的约束，即

$$X=(k_1,k_2,k_3,c_1,c_2,c_3,d)^{\mathrm{T}}$$

（4）建立动态响应优化数学模型

动态响应优化数学模型为

$$\begin{cases}\overline{\varphi}_0=d\\ \left|\ddot{y}_1(t)\right|\leqslant d\\ \begin{Bmatrix}\dot{x}\\ \dot{z}\end{Bmatrix}+\begin{bmatrix}0 & -\boldsymbol{I}_{5\times5}\\ \boldsymbol{K} & \boldsymbol{C}\end{bmatrix}\begin{Bmatrix}x\\ z\end{Bmatrix}=\begin{Bmatrix}0\\ \boldsymbol{b}F(t)\end{Bmatrix}\\ \left|y_2(t)+\frac{L}{12}y_3(t)-y_1(t)\right|\leqslant\theta_1\\ \left|y_4(t)-y_2(t)-\frac{L}{3}y_3(t)\right|\leqslant\theta_2\\ \left|y_5(t)-y_2(t)-\frac{2L}{3}y_3(t)\right|\leqslant\theta_3\\ \left|y_4(t)-f_1(t)\right|\leqslant\theta_4,\left|y_5(t)-f_2(t)\right|\leqslant\theta_5\\ x_{\min}\leqslant x\leqslant x_{\max},z_{\min}\leqslant z\leqslant z_{\max},d\geqslant0\end{cases} \tag{3.58}$$

1）汽车悬架系统动态响应优化设计问题一（第一种路况）

第一种路面轮廓如图 3.26 所示。汽车速度 v=11.43m/s。路面激励频率为

$\omega_i=\dfrac{\pi v}{L_i}$，$L_1$=9.144m，$L_2$=3.6576m，后轮到达前轮未知的时间间隔为 $t_\sigma=0.2667\text{s}$，其中路面起伏不平所激起前后轮的垂直位移函数 $f_1(t)$ 及 $f_2(t)$ 分别为

$$f_1(t)=\begin{cases}y_0[1-\cos\omega_i(t-t^{i-1})],t^{i-1}\leqslant t\leqslant t^i,i=1,3,\cdots,2n-1\\ y_0[1+\cos\omega_i(t-t^{i-1})],t^{i-1}\leqslant t\leqslant t^i,i=1,3,\cdots,2n\end{cases}\quad 0\leqslant t\leqslant t_1 \tag{3.59}$$

$$f_2(t)=f_1(t-t_\sigma),t_\sigma\leqslant t\leqslant t_1+t_\sigma \tag{3.60}$$

式中，$\omega_1=1.25\pi\,\text{rad/s}$，$\omega_2=3.125\pi\,\text{rad/s}$。

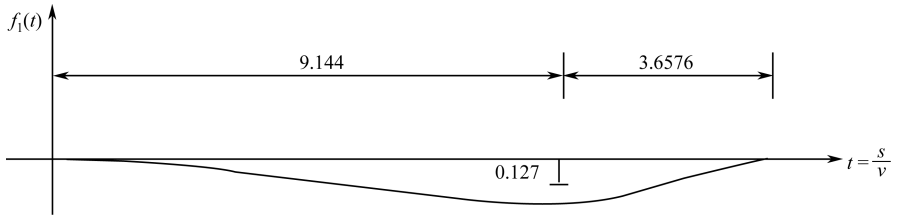


图 3.26　第一种路面轮廓

将第一种路况函数 $f_1(t)$ 和 $f_2(t)$，即式（3.59）和式（3.60）代入式（3.49）应用谱元法求解，结果如图 3.27 和图 3.28 所示。两种方法的目标函数迭代过程比较如图 3.29 所示。

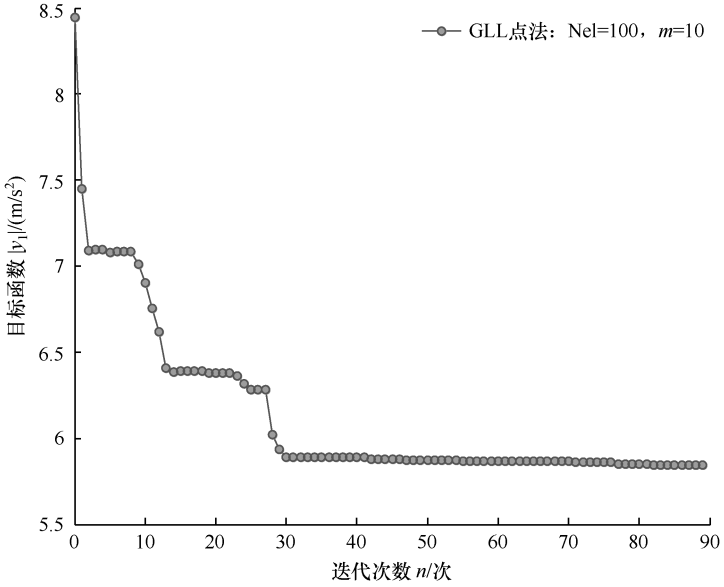


图 3.27　Nel=100，m=10 时 GLL 点法的目标函数迭代过程（第一种路况）

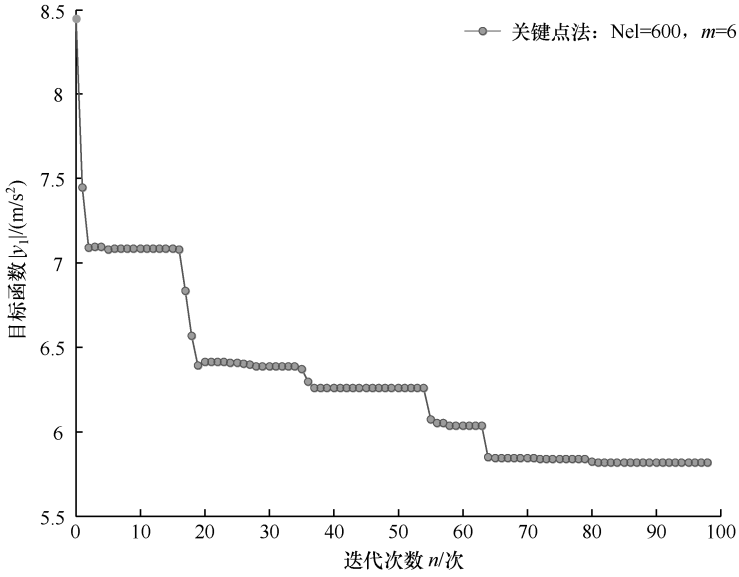


图 3.28　Nel=600，m=6 时关键点法的目标函数迭代过程（第一种路况）

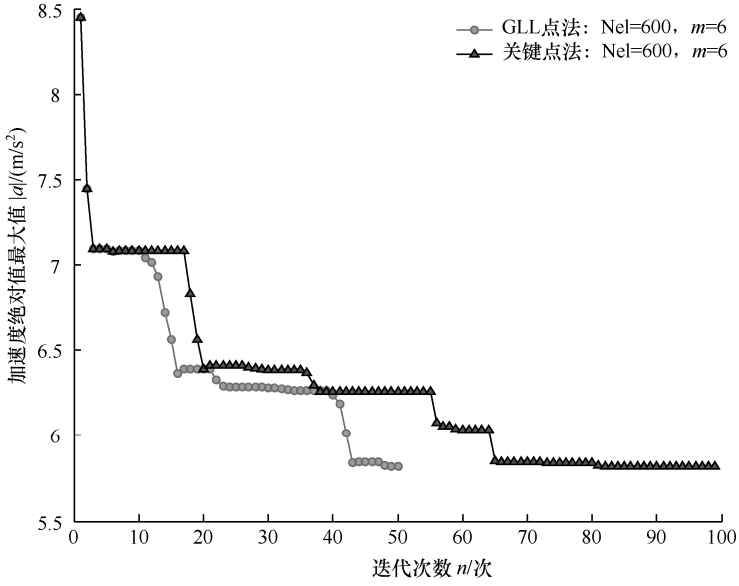


图 3.29　Nel=600，m=6 时两种方法的目标函数迭代过程比较（第一种路况）

在本节的 1、2 中，分别对单自由度和两自由度物理模型系统进行了优化，结论为 GLL 点法对于处理与时间有关的约束的多自由度动态问题，在耗费时间上有优越性；关键点法对于处理与时间有关的约束的多自由度动态问题，在准确性方面有优越性。因此，在本节的 3 中进行多自由度物理模型系统研究时，分别采用

了 GLL 点法和关键点法，GLL 点法的结果如表 3.19 和表 3.20 所示。当单元数 Nel=10，插值点数 m=3 时，可得到较好的结果，迭代 70 次耗时仅为 89s；当单元数 Nel=20，插值点数 m=6 时，迭代 200 次耗时 501s，得到的目标值为 5.765。由此可以看出，并不是单元数和插值点数越多越好。

表 3.19　GL L 点法的数值试验最优点（问题一）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	k_1	8756.3	8756.3	8756.3	11512.577	11313.764	11368.397
		k_2	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	56096.965	52896.294	54380.17	54236.69	54012.453	53880.53
		c_1	5874.6508	5335.082	7439.2012	5311.5278	8313.5847	8672.8911
		c_2	8698.1375	8683.9768	8744.6034	8732.3245	8751.1485	8751.0193
		c_3	875.63	875.63	875.63	875.63	875.63	875.63
		d	5.635387	5.679881	5.820114	5.861439	5.842738	5.842372
	6	k_1	16829.493	8756.3	11375.591	11147.023	11313.574	11190.163
		k_2	52441.299	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	52444.881	52325.981	54058.037	53953.222	53887.454	54057.191
		c_1	8369.2674	8756.3	8595.8982	8460.695	8690.769	8739.3201
		c_2	6921.9346	8711.5346	8752.9325	8750.414	8751.0811	8754.734
		c_3	3742.2472	875.63	875.63	875.63	875.63	875.63
		d	6.391697	5.783691	5.840103	5.840892	5.841976	5.840749
	10	k_1	10463.497	10988.386	11465.389	11354.467	10781.71	11333.207
		k_2	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	53007.71	54250.642	54200.764	53769.299	54068.151	54080.893
		c_1	6989.117	7134.3363	5752.0915	8756.2999	8756.2999	8682.4604
		c_2	8717.1598	8730.6553	8738.76	8748.2939	8754.5056	8755.1213
		c_3	875.63	875.63	875.63	875.63	875.63	875.63
		d	5.784874	5.826069	5.858013	5.842045	5.837389	5.842109

			插值点数 m		
			3	6	10
单元数 Nel	400	k_1	10621.090976	11319.587419	11420.985152
		k_2	35025.200000	35025.200000	35025.200000
		k_3	54296.810450	54167.781032	53796.788090
		c_1	8749.225365	8756.300000	8756.300000
		c_2	8758.980864	8757.287613	8749.721132
		c_3	875.630000	875.630000	875.630000
		d	5.836000	5.841821	5.842758

（续表）

			插值点数 m		
			3	6	10
单元数 Nel	600	k_1	11352.775167	8756.314798	16119.914150
		k_2	35025.200000	35025.200000	50497.415772
		k_3	54092.351775	54303.524191	52197.682887
		c_1	8355.953880	8724.263448	8756.300000
		c_2	8753.773509	8756.446953	8822.499714
		c_3	875.630000	875.630000	875.630000
		d	5.843105	5.820047	6.285569
	800	k_1	11304.571321	10852.478996	11346.898801
		k_2	35025.200000	35025.200000	35025.200000
		k_3	54015.669859	54105.030017	53926.860607
		c_1	8594.349953	8756.300000	8756.300000
		c_2	8753.261642	8755.382237	8752.349274
		c_3	875.630000	875.630000	875.630000
		d	5.842101	5.837981	5.842124

表 3.20　GLL 点法的数值试验的 CPU 耗时（s）与迭代次数（问题一）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	t	21.3125	32.9375	55.34375	45.5625	111.39063	105.03125
		n	100	134	94	48	63	46
	6	t	7.875	33.59375	59.859375	158.04688	253.07813	226.28125
		n	30	73	50	81	72	48
	10	t	30.03125	58.65625	155.89063	374.625	737.8125	692.375
		n	64	69	60	90	96	55
			插值点数 m					
			3		6		10	
单元数 Nel	400	t	235.062500		462.109375		1217.343750	
		n	77		64		71	
	600	t	286.750000		569.921875		1217.421875	
		n	68		50		46	
	800	t	305.218750		1251.312500		2657.671875	
		n	56		70		72	

关键点法的结果如表 3.21 和表 3.22 所示。除了单元数 10、20，插值数 3、6、10 外，其他情况都获得了满意的结果，然而耗时都非常多，最多的是 49741.078125s，

即 13.8170h，单元数和插值数分别为 600 和 10，迭代次数为 500。因此从准确性来说，关键点法有优势，但是耗时特别多，在本例中竟然耗时 13.8170h，对于复杂的系统耗时会急剧增加，甚至无法容忍。

表 3.21　关键点法的数值试验最优点（问题一）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	k_1	8756.3000	8756.3000	8756.3000	8756.3000	8756.3000	8756.3000
		k_2	35025.200	35025.200	35025.200	35025.200	35025.200	35025.200
		k_3	50346.570	35025.200	35025.200	50358.411	52366.735	52807.800
		c_1	8756.300	1676.1812	3645.4055	8682.0458	8266.9709	8173.8100
		c_2	13133.195	8216.2125	7735.2723	8613.9762	8680.5405	8712.3996
		c_3	875.63000	9454.7825	1257.1622	875.63000	875.63000	875.63000
		d	1.963638	3.910859	5.478822	5.698961	5.787683	5.805585
	6	k_1	8756.3000	26698.367	8756.3000	8756.3000	8756.3000	8756.3000
		k_2	35025.200	35025.200	35025.200	35025.200	35025.200	35025.200
		k_3	35025.200	35025.200	39705.641	35025.200	52388.835	52907.790
		c_1	8756.3000	1000.4292	8756.3000	3943.0067	8268.6120	8174.8995
		c_2	14010.080	7602.9390	8093.1504	8081.0408	8681.0326	8714.5676
		c_3	4202.5142	9525.4147	875.63000	997.60482	875.63000	875.63000
		d	1.925773	4.018860	5.593787	5.669711	5.787600	5.805577
	10	k_1	8756.3000	28163.175	8756.3000	8756.3000	8756.3000	8756.3000
		k_2	35025.200	35025.200	35025.200	35025.200	35025.200	35025.200
		k_3	35025.200	35025.200	35025.200	50571.908	52409.528	53301.292
		c_1	8756.3000	930.22129	3631.4489	8675.3064	8268.8297	8179.3519
		c_2	14010.080	7394.7033	7744.7271	8619.1501	8681.4695	8723.0446
		c_3	4281.7080	6917.9697	1242.5194	875.63000	875.63000	875.63000
		d	1.956334	4.006835	5.484301	5.698983	5.787598	5.805534

			插值点数 m		
			3	6	10
单元数 Nel	400	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	52798.648129	52879.662203	52827.155084
		c_1	6342.630820	8145.481357	8144.903038
		c_2	8702.972389	8717.845383	8716.731020
		c_3	875.630000	875.630000	875.630000
		d	5.807309	5.811567	5.811576
	600	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000

（续表）

			插值点数 m		
			3	6	10
单元数 Nel	600	k_3	53765.972970	53689.828180	64340.450777
		c_1	8134.424942	8133.434345	8401.268680
		c_2	8740.327862	8738.745807	8928.444706
		c_3	875.630000	875.630000	875.630000
		d	5.815653	5.815670	5.811778
	800	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	54195.095496	54186.287798	54218.661737
		c_1	7127.824714	8132.720909	8133.165384
		c_2	8743.330250	8750.008370	8750.678298
		c_3	875.630000	875.630000	875.630000
		d	5.814718	5.817054	5.817046

表 3.22　关键点法的数值实验的 CPU 耗时（s）与迭代次数（问题一）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	t	275.890	547.625	3702.125	1553.796	3181.859	6046.281
		n	104	142	404	110	121	161
	6	t	463.031	959.312	4899.640	1687.250	4034.890	6417.500
		n	195	233	500	111	139	154
	10	t	475.406	789.125	2184.203	1512.609	3494.187	4649.609
		n	167	143	186	82	101	92
			插值点数 m					
			3		6		10	
单元数 Nel	400	t	5639.187500		9405.093750		11047.390625	
		n	116		171		168	
	600	t	12152.203125		7978.203125		49741.078125	
		n	166		99		500	
	800	t	10085.734375		10823.031250		39162.531250	
		n	108		98		293	

本节结果与文献的比较如表 3.23 所示。

2）汽车悬架系统动态响应优化设计问题二（第二种路况）

路面轮廓的位移函数 $f_1(t)$、$f_2(t)$ 如图 3.30 和图 3.25 所示。汽车速度 v=24.384m/s。这样 t_σ=0.125s 和两个环境参数 ω_i=2πrad/s 和 16πrad/s（i=1，2，3，4）适合于两种路面。

表 3.23　本节结果与文献的比较

	本节			文献[63]	
	初始值	最优值		初始值	最优值
		GLL 点法	关键点法		
k_1	17512.6	11375.60	8756.30	17512.6	8756.30
k_2	52537.8	35025.20	35025.20	52537.8	35025.25
k_3	52537.8	54058.03	52879.66	52537.8	42362.98
c_1	1751.26	8595.90	8145.48	1751.26	2257.37
c_2	4378.15	8752.93	8717.84	4378.15	13575.77
c_3	4378.15	875.63	875.63	4378.15	14010.08
d（目标函数）	8.448	5.840	5.811567	8.448	6.538

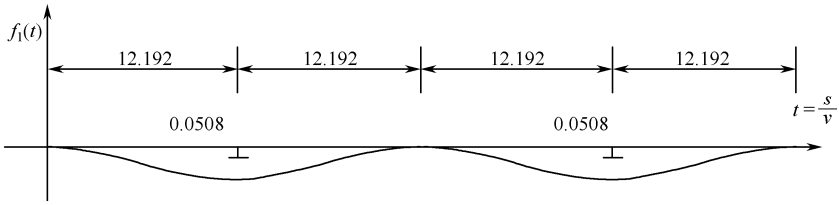


图 3.30　第二种路面轮廓

将第二种路况函数 $f_1(t)$ 和 $f_2(t)$ 代入式(3.49)应用谱元法求解，结果如图 3.31～图 3.33，表 3.24～表 3.28 所示。

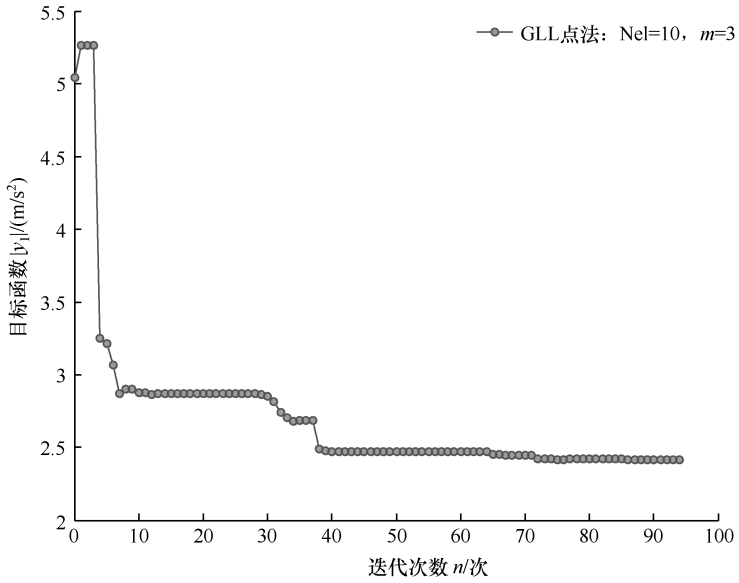


图 3.31　Nel=10，m=3 时 GLL 点法的目标函数迭代过程（第二种路况）

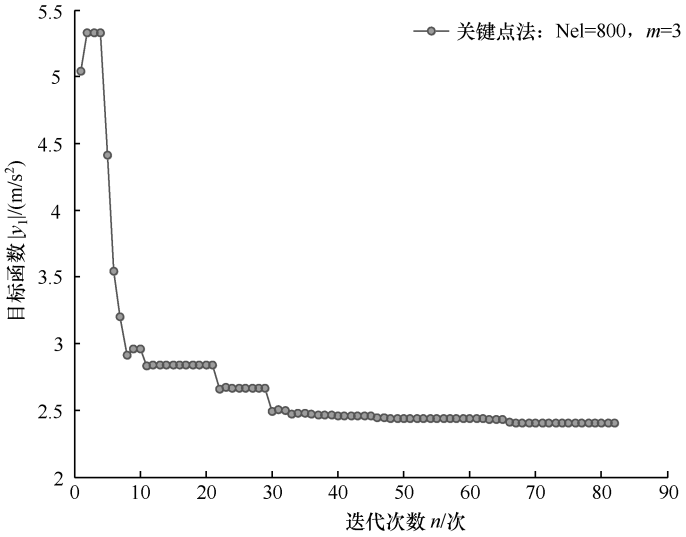


图 3.32　Nel=800，m=3 时关键点法的目标函数迭代过程（第二种路况）

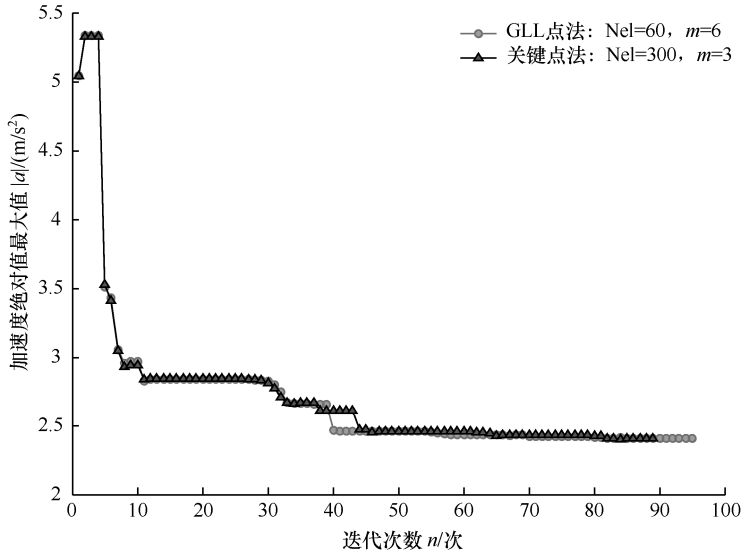


图 3.33　两种方法目标函数迭代过程比较（第二种路况）

表 3.24　GL L 点法的数值试验最优点（问题二，ω_i=2πrad/s）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	k_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		k_2	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2

（续表）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	c_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		c_2	14010.08	14010.08	14010.08	14010.08	14010.08	14010.08
		c_3	12025.239	14010.08	12405.143	11886.817	12365.499	12006.002
		d	2.415443	2.389652	2.404458	2.404392	2.404489	2.404502
	6	k_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		k_2	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		c_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		c_2	14010.08	14010.08	14010.08	14010.08	14010.08	14010.08
		c_3	10706.198	11168.851	12469.341	11938.339	12161.315	12264.579
		d	2.399473	2.402319	2.404396	2.404432	2.404494	2.404543
	10	k_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		k_2	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		k_3	35025.2	35025.2	35025.2	35025.2	35025.2	35025.2
		c_1	8756.3	8756.3	8756.3	8756.3	8756.3	8756.3
		c_2	14010.08	14010.08	14010.08	14010.08	14010.08	14010.08
		c_3	12239.535	11353.901	11964.014	12087.739	12231.532	12160.188
		d	2.40455	2.402336	2.40447	2.404544	2.404552	2.404547
			插值点数 m					
			3		6		10	
单元数 Nel	400	k_1	8756.300000		8756.300000		8756.300000	
		k_2	35025.200000		35025.200000		35025.200000	
		k_3	35025.200000		35025.200000		35025.200000	
		c_1	8756.300000		8756.300000		8756.300000	
		c_2	14010.080000		14010.080000		14010.080000	
		c_3	12214.538067		12205.037649		12162.990727	
		d	2.404555		2.404557		2.404558	
	600	k_1	8756.300000		8756.300000		8756.300000	
		k_2	35025.200000		35025.200000		35025.200000	
		k_3	35025.200000		35025.200000		35025.200000	
		c_1	8756.300000		8756.300000		8756.300000	
		c_2	14010.080000		14010.080000		14010.080000	
		c_3	12137.601981		12214.828887		12194.592986	
		d	2.404549		2.404555		2.404557	

（续表）

			插值点数 m		
			3	6	10
单元数 Nel	800	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	35025.200000	35025.200000	35025.200000
		c_1	8756.300000	8756.300000	8756.300000
		c_2	14010.080000	14010.08000	14010.080000
		c_3	12106.941020	12191.425283	12191.426661
		d	2.404546	2.404558	2.404558

表 3.25　GLL 点法的数值试验的 CPU 耗时和迭代次数（问题二，ω_i=2πrad/s）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	t	19.921875	14.14062	54.265625	58.046875	193.60938	286.60938
		n	95	60	95	67	115	125
	6	t	25.625	44.53125	108.70313	153.71875	268.53125	631.10938
		n	97	102	95	85	77	121
	10	t	31.046875	66.25	204.0625	342.73438	691.42188	1059.5781
		n	70	81	87	84	91	92
			插值点数 m					
			3		6		10	
单元数 Nel	400	t	214.687500		634.421875		2263.687500	
		n	75		84		142	
	600	t	509.296875		880.828125		2509.687500	
		n	130		85		101	
	800	t	415.531250		1571.234375		4124.843750	
		n	82		90		119	

表 3.26　关键点法数值试验最优值（问题二，ω_i=2πrad/s）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	k_1	8756.300	8756.300	8756.300	8756.300	8756.300	8756.300
		k_2	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		k_3	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		c_1	6433.267	8464.265	8756.300	8756.300	8756.300	8756.300
		c_2	6968.144	9774.998	12976.43	13828.46	14010.08	14010.08
		c_3	5638.458	7520.940	9844.320	10472.06	11176.12	11706.51
		d	1.513167	2.012354	2.363677	2.388839	2.402354	2.404106
	6	k_1	8756.300	8756.300	8756.300	8756.300	8756.300	8756.300

（续表）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	6	k_2	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		k_3	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		c_1	6721.638	8494.505	8756.300	8756.300	8756.300	8756.300
		c_2	6870.165	9765.838	12976.41	13828.46	14010.08	14010.08
		c_3	5483.822	7532.136	9844.383	10472.08	11176.14	11706.48
		d	1.530744	2.012644	2.363686	2.388839	2.402354	2.404106
	10	k_1	8756.300	8756.300	8756.300	8756.300	8756.300	8756.300
		k_2	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		k_3	35025.20	35025.20	35025.20	35025.20	35025.20	35025.20
		c_1	6721.393	8494.492	8756.300	8756.300	8756.300	8756.300
		c_2	6870.271	9765.837	12976.41	13828.46	14010.08	14010.08
		c_3	5483.937	7532.144	9844.395	10472.07	11176.09	12864.02
		d	1.530754	2.012644	2.363686	2.388839	2.402354	2.403709

			插值点数 m		
			3	6	10
单元数 Nel	400	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	35025.200000	35025.200000	35025.200000
		c_1	8756.300000	8756.300000	8756.300000
		c_2	14010.080000	14010.080000	14010.080000
		c_3	12861.782791	12861.769793	12861.847829
		d	2.403714	2.403714	2.403714
	600	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	35025.200000	35025.200000	35025.200000
		c_1	8756.300000	8756.300000	8756.300000
		c_2	14010.080000	14010.080000	14010.080000
		c_3	11823.005867	11823.005350	11823.005543
		d	2.404107	2.404107	2.404107
	800	k_1	8756.300000	8756.300000	8756.300000
		k_2	35025.200000	35025.200000	35025.200000
		k_3	35025.200000	35025.200000	35025.200000
		c_1	8756.300000	8756.300000	8756.300000
		c_2	14010.080000	14010.080000	14010.080000
		c_3	12403.231285	11985.009577	11984.975934
		d	2.404460	2.404487	2.404487

表 3.27　关键点法数值试验 CPU 耗时和迭代次数（问题二，ω_i=2πrad/s）

			单元数 Nel					
			10	20	60	100	200	300
插值点数 m	3	t	356.656	563.468	1452.187	1219.937	3391.265	3411.593
		n	150	145	157	85	126	89
	6	t	410.531	462.093	1286.687	1824.187	2954.875	2457.953
		n	171	112	130	117	100	58
	10	t	455.328	602.281	1868.968	1626.234	2922.296	2696.046
		n	164	125	157	87	83	53

		插值点数 m		
		3	6	10
单元数 Nel	400	6874.562500	4558.109375	5946.828125
		141	86	92
	600	8408.000000	5869.468750	7520.703125
		116	72	75
	800	7289.296875	8295.234375	14872.359375
		82	79	117

表 3.28　本节结果与文献的比较

	本节			文献[63]	
	初始值	最优值		初始值	最优值
		GLL 点法	关键点法		
k_1	17512.6	8756.30	8756.30	17512.6	8756.30
k_2	52537.8	35025.20	35025.20	52537.8	35025.25
k_3	52537.8	35025.20	35025.20	52537.8	35025.25
c_1	1751.26	8756.30	8756.30	1751.26	1563.88
c_2	4378.15	14010.080	14010.08	4378.15	8041.79
c_3	4378.15	12214.82	11823.00	4378.15	6621.51
d（目标函数）	5.044	2.405	2.404	5.044	3.188

机械系统的动态响应优化设计有很好的应用前景[70]。动态响应必须满足依赖于时间的微分方程。为了获得最优解且满足与时间有关的约束，要求获得整个时间上的响应。本章应用谱元法计算了整个时间内的响应，将微分方程或方程组转化为代数方程组；应用序列二次规划（SQP）优化算法进行了优化计算。对于与时间有关的约束的处理，采用了两种方法，即关键点法和 GLL 点法。通过分析线性单自由度减振器设计问题，线性两自由度减振器设计问题和 5 自由度汽车悬架系统设计问题，验证了基于谱元法的动态响应优化的可行性。当采用 GLL 点法处

理与时间有关的约束的多自由度动态问题时，在时间上有优越性，但要求较多的单元数；关键点法在处理与实践有关的约束的多自由度动态问题时，在准确性上有优越性，虽然采用了很少的单元，但是在每个单元内要对高次 Lagrange 函数寻找基于时间的最优值，寻优的次数是 2×（状态变量数+状态变量的组合数），则在整个优化过程中，每迭代一步，对高次 Lagrange 函数寻找基于时间的最优值的次数等于单元数×2（状态变量数+状态变量的组合数）。例如，单元数是 6，状态变量数是 2，状态变量组合数为 1，则在一次迭代中要寻优 36 次，假如迭代 10 次，则要寻优 360 次，这将非常耗时，因此关键点法没有优势，本章的算例也证明了这一点；但是关键点法能找到最优解的概率更大。从表 3.20 中可以看出，当单元数在 6 及 6 个以上时，都找到了合理的结果，即关键点法的结果波动很小。优化结果波动比较大是由于 GLL 点不在最优值点附近造成的。在 3.5.2 节的 3．中，针对 5 自由度汽车悬架系统的动态响应优化设计问题采用了 GLL 点法和关键点法进行分析计算，说明了在多自由度动态响应优化问题中，为获得更好的效果，增加单元数和插值点数是可行的，但是当单元数和插值点数增加到一定数量时，再增加对结果没有影响，即说明已达到足够的精度。两自由度减振器设计问题代表最简单的多自由度动力学设计问题，5 自由度汽车悬架系统设计问题代表比较复杂的多自由度动力学设计问题。对于机械零部件的动态响应优化设计问题及动载荷弹性分布参数系统的优化设计，可以参考本章的方法解决，如在固定端承受振动输入的矩形变截面梁的动态响应优化设计，在不同边界条件下承受垂直平面均布瞬态动载荷作用的弹性梁的动态响应优化设计等。

3.6　本章小结

本章研究了基于时间谱元法的系统动态响应优化算法，深入分析了在时间域内的离散动态响应，并将运动微分方程组转化成代数方程组，精确解出了瞬态响应，用 GLL 点法和关键点法处理了时间约束。本章以线性单自由度减振器设计，线性两自由度减振器设计和 5 自由度汽车悬架系统设计为例，引入人工设计变量，详细研究了两种处理约束方法（关键点法和 GLL 点法）的优缺点，也说明了 GLL 点法和关键点法的正确性。本章的内容可为进一步研究动态响应优化提供参考，如在此基础上研究复杂系统的灵敏度分析，以提高本章提出的两种方法的实用性等。

第 4 章　多元模型自适应与时间谱元法结合的动态响应优化

机械结构几乎都在动态载荷的环境下工作，其各种性能都是依赖于时间的函数。为了提升机器的动态性能，动态响应优化是非常必要的。而动态响应优化需要处理载荷在时间域上的变化，精确求出响应在时间域上的解，并且要满足所有依赖于时间的约束函数，不能漏掉时间域上的响应，特别是绝对极值点。动态响应优化器在每一步迭代中，都需要解依赖于时间的目标函数与约束函数，并进行灵敏度分析，因此，在动态响应优化过程中，要求精确、高效地获得依赖于时间的响应。

求解动态响应的方法常用的是有限差分法，它是用很小的时间步长进行计算的，计算的稳定性与准确性依赖于时间步长的大小，当时间步小时，计算非常耗时；当时间步大时，求解响应精度低，动态响应优化不稳定或不收敛。特别是当结构受到冲击载荷时，有限差分法更无能为力。然而，文献[17]对受冲击载荷的结构应用谱元法获得了非常精确的动态响应，并且达到了谱收敛，而没有增加单元的数量。

在 min-max 问题中，最大响应是随着优化过程变化的，因此直接对目标函数进行优化，得不到最优解或发散[11]，Dong-Hoon Choi 等提出设置一个目标函数的绝对极值响应系数 $\alpha(0<\alpha\leqslant 1)$，并且将目标函数转化为一个约束，即小于等于绝对极值响应系数。文献[71]采用时间谱元法离散结构动力学方程的时间，应用 GLL 点法和关键点法处理依赖于时间的约束，说明了 GLL 点法不稳定，对单元数和插值次数比较敏感，而关键点法由于每一次迭代的最大值的位置发生变化，所以导致收敛速度非常慢甚至发散。

19 世纪 70 年代，人们才开始研究承受瞬态载荷的结构动态响应优化[72]。2006 年，Kang 等首次将动态响应优化当作一个优化的分支[18]。处理与时间相关约束的方法有等价函数法[73]、极端情况法[74]、局部极大响应法[75]、局部极大响应附近逐点法[76]和网格点法[77]。当应用等价函数法时，与时间相关的约束用一个等价函数表示，且 KKT（Karush-Kuhn-Tucker）必要条件一般是不相同的。极端情况法仅仅把被激活约束的最大响应作为一个约束，收敛很慢或发散。为了克服极端情况法的缺点，考虑将所有局部最大响应作为约束。当存在局部最大响应时，需要额外努力找到它们对应的时间点。处理与时间相关约束的最简单的措施是考虑应用

激活约束集策略，即在所有时间网格点的响应约束。如果一个约束在大部分时间网格点都没有违约，则采用网格点法具有适应性好、简单易行的优点，但是其约束数量多，耗费巨大资源，且难以保证网格之间的点的动态响应满足约束。

秦仙蓉等[78]采用分层的思想对动态优化问题进行了整体层和局部层的动态优化。马斌[79]从瞬态分析结果中取出最大应力的时间点对应的结果，将其作为静态工况对连杆进行了优化设计，其实质是静态优化。杜雪松等[80]以齿轮的振动加速度和质量为目标函数，对船用齿轮传动进行了多目标动态优化，其目标函数是依赖于时间的函数。王晓博等[81]以转子质量和不平衡振动响应为状态变量，以转子的固有频率为目标函数对结构进行了动态优化设计。乔崇全等[82]详细叙述了对大型圆振动筛进行的动力学分析。杨斌等[83]提出了一种基于动态响应的变约束迭代优化方法，解决了静态响应的优化结果往往不满足动态响应约束的问题。王志强等[84]采用径向基函数插值方法对动态优化问题进行了全离散，得到了近似的大规模非线性规划问题。史洪岩等[85]详细综述了动态优化方法的最新发展，而且提出了动态优化进一步的研究方向。总之，学者们研究动态优化问题时，仅仅从处理一个具体的动态优化问题且获得比以前更优的解的角度来思考问题，而很少从一般的动态响应优化问题入手，对它的三个方向（动力学方程的求解、优化算法及依赖于时间的约束处理）进行研究。本章对这三个研究方向分别进行了研究。

随着计算机仿真技术的发展，工程设计师们把目标都聚焦在通过仿真技术来认识问题，应用仿真优化来改造问题。优化技术包括两类：一类是基于梯度的优化，如序列二次规划法等；另一类是仿生算法，如遗传算法。在优化过程中，基于梯度的优化需要应用中心差分法计算梯度信息，每迭代一次需要仿真 $2n+1$ 次[86]，而仿真通常比较耗时。仿生算法尽管不需要计算梯度信息，但是需要通过种群不断演化，需要进行大量仿真[87]，而且仿生算法没有收敛的概念，是通过设置演化次数来停止算法的，因此仿生算法的效率更低。元模型，也就是“模型的模型”，即代理模型，可以代替精确模型进行仿真优化，大大节约了成本。目前，比较成熟的元模型有多变量自适应回归样条（Multivariate Adaptive Regression Splines，MARS）[88]、径向基函数（RBF）和克利金模型（kriging）[89]等。研究最多的基于元模型的优化方法是模式追踪算法（Mode Pursuing Sampling，MPS）[90]、高效全局优化算法（Efficient Global Optimization，EGO）[87]和元模型混合自适应方法（Hybrid and Adaptive Metamodeling Method，HAMM）[91]等。

本章提出关键点集法处理与时间相关的约束，并使函数绝对值最大的点附近的高斯-罗巴托-勒让德（Gauss-Lobatto-Legendre，GLL）点被包括在约束集中，克服当前迭代的局部最大值可能在下一次迭代不是局部绝对最大值的缺点；提出元模型混合自适应方法及处理与时间相关的约束，采用时间谱元法计算动态响应及

与时间相关的约束，提取绝对极值最大的约束函数的值及其对应的设计变量，通过这些数据构建混合元模型，然后应用多变量自适应回归样条、径向基函数和克利金模型这三个元模型来决定采样点的取舍，删除不满足约束元模型的采样点。

4.1 时间谱元法

1984 年，Patera 提出了谱元法[16]，它首先在流体动力学中得到了应用，融合了有限元法的处理结构边界的灵活性与谱方法的快算收敛的优点。在相同精度的前提下，谱元法仅仅应用了较小的单元，且在每个单元中，把时间离散为与 GLL 点相对应的插值点，对这些点进行 Lagrange 插值。

理论上，当点的数量不变时，在正交多项式的零点上插值，能得到插值精度最高的插值函数[59]。在时间均匀点处和 GLL 点处分别插值来近似龙格函数 Runge，如图 4.1 所示，对于均匀离散的近似，当插值点数提高时，近似函数的误差非常大或面目全非，特别是在靠近设计区域的边界处；而 GLL 点的插值非常明显地接近原始函数。GLL 点的分布如图 4.2 所示。

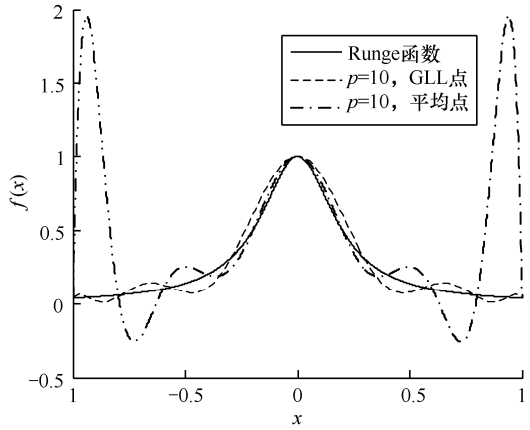


图 4.1　在 GLL 点和平均点处分别进行 Lagrange 插值来近似 Runge 函数

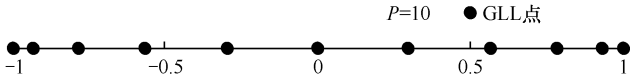


图 4.2　GLL 点的分布（p 表示单元内的插值点数，在[−1,1]区间上）

时间谱元法的主要步骤为：首先转化二阶微分方程组为一阶微分方程组；其次将时间区域划分成若干谱单元，每个单元由若干个谱节点组成；再次用 Legendre 正交多项式作为基函数，将每一个单元的近似解表达为基函数的线性组合；接下

来应用 Bubnov-Galerkin 原理转化为等效的积分形式，构成单元谱元方程；再通过连接矩阵将所有单元的谱元方程叠加成总体谱元方程；最后解时间域上的总体谱元方程，求出时间域上的状态变量的全局近似解。

4.1.1　结构动力学方程及其转化形式

结构动力学方程可表达为

$$\boldsymbol{M}\ddot{x}+\boldsymbol{C}\dot{x}+\boldsymbol{K}x=F \tag{4.1}$$

式中，$\boldsymbol{M}$ 为质量矩阵；$\boldsymbol{C}$ 为阻尼矩阵；$\boldsymbol{K}$ 为刚度矩阵；F 为时间函数载荷向量；x 为位移向量；$\ddot{x}$ 为加速度向量；$\dot{x}$ 为速度向量，初始条件为 $x(0)=b_0$，$\dot{x}(0)=v_0$。

$\boldsymbol{M}$，$\boldsymbol{C}$，$\boldsymbol{K}$ 不随时间变化；x，$\ddot{x}$，$\dot{x}$ 为时间的函数；F 为时间的任意函数，$t\in[t_0,t_n]$。

设 $x_1=\dot{x}$，$x_2=x$，将式（4.1）转化为一阶线性常微分方程组式（4.2）：

$$\begin{Bmatrix}\dot{x}_1\\ \dot{x}_2\end{Bmatrix}+\begin{bmatrix}\boldsymbol{M}&0\\0&\boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{C}&\boldsymbol{K}\\-\boldsymbol{I}&0\end{bmatrix}\begin{Bmatrix}x_1\\x_2\end{Bmatrix}=\begin{bmatrix}\boldsymbol{M}&0\\0&\boldsymbol{I}\end{bmatrix}^{-1}\begin{Bmatrix}F\\0\end{Bmatrix}$$
$$\{x_1(0)\}=\{x_1^0\},\{x_2(0)\}=\{x_2^0\} \tag{4.2}$$

4.1.2　单元划分

如图 4.3 所示，将仿真时间 $t\in[t_0,t_n]$ 离散为 n 个互不相交的谱单元，即 $[t_0,t_1]$，$[t_1,t_2]$，$[t_2,t_3]$，…，$[t_{n-1},t_n]$，对每单元配置若干个点。

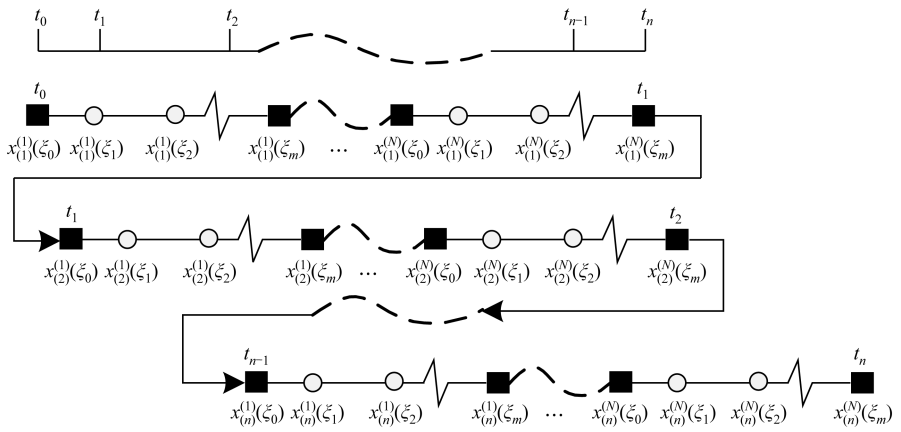


图 4.3　用谱元法离散时间

4.1.3 单元分析

将时间离散为谱单元以后，通过在单元内添加插值点提高数值解的精度，这在有限元中称为 hp 方法。如果将单元内的插值点离散在 GLL 点上，可以获得谱收敛，即指数级收敛，并且精度也非常高。GLL 点是 GLL 多项式的零点。

在每一个单元内，将状态变量近似为 m 次 Lagrange 函数：

$$\tilde{x}^{(j)}(\xi)=\sum_{k=0}^{m}x^{(j)}(\xi_k)P_k^{(j)}(\xi) \tag{4.3}$$

式中，$P_k^{(j)}(\xi)$ 为 j 单元的 k 次 Lagrange 多项式；ξ_k 为定义在[−1,1]上的 GLL 点；$x^{(j)}(\xi_k)$ 为单元 j 上未知节点在 GLL 点的值。

将式（4.2）展开，其中一个方程可以表达为

$$\begin{aligned}&\dot{x}+\boldsymbol{A}_s x=f(t)\\&x_0=d_{i0},\dot{x}_0=v_{i0},t\in[t_i,t_{i+1}]\end{aligned} \tag{4.4}$$

式中，$\boldsymbol{A}_s$ 为耦合矩阵。

把式（4.3）代入式（4.4），对于每一个单元，采用 Bubnov-Galerkin 法[60]保证插值误差最小，得到

$$\sum_{j=1}^{\mathrm{Nel}}\int_{-1}^{1}P_n^{(j)}\left[\frac{\mathrm{d}\tilde{x}^{(j)}}{\mathrm{d}\xi}+\frac{h^{(j)}}{2}\{\boldsymbol{A}_s\tilde{x}^{(j)}-f^{(j)}(\tilde{x}^{(j)},\xi)\}\right]\mathrm{d}\xi=0 \tag{4.5}$$

将式（4.5）中的每一个单元用矩阵形式表示为

$$\boldsymbol{L}^e x^e(t)=\boldsymbol{F}^e(t) \tag{4.6}$$

式中，$\boldsymbol{L}^e=\boldsymbol{\Phi}-\boldsymbol{A}_s I_\omega^{(e)}$；$x^e(t)=\left(\left[x(\xi_0)\ \ x(\xi_1)\ \cdots\ x(\xi_m)\right]^{(e)}\right)^{\mathrm{T}}$；$\boldsymbol{F}^e(t)=-I_\omega^{(e)}f^{(e)}$，$I_\omega$ 为 ξ 的一般函数。

4.1.4 集成总体时间谱元方程并求解

总体集成就是把所有单元的谱元方程通过连接矩阵组合起来，得到总体时间谱元方程，即线性方程组。对于 N_v^i 个状态变量，全部状态变量的全局形式是通过耦合矩阵 $\boldsymbol{A}_s^i$（$N_v^i\times N_v^i$ 的方阵）的张量叉乘获得的：

$$(I^i\otimes\boldsymbol{B}_u^i)X_{ug}^i=(\boldsymbol{A}_s^i\otimes\boldsymbol{B}_w^i)X_{ug}^i-(\boldsymbol{I}^i\otimes\boldsymbol{B}_w^i)F_{ug}^i(X_{ug}^i) \tag{4.7}$$

式中，$\boldsymbol{B}_u^i$、$\boldsymbol{B}_w^i$ 为第 i 个谱单元的全局微分矩阵与全局权矩阵；$F_{ug}^i(X_{ug}^i)$ 为第 i 个谱单元所受激励力的全局形式；$\boldsymbol{I}^i$ 为 $N_v^i\times N_v^i$ 单位矩阵；X_{ug}^i 为第 i 个谱单元的所有状态变量在时间节点上的组合。

化简式（4.7）得到：

$$\boldsymbol{G}^i X_{ug}^i = -\boldsymbol{B}_{ug}^i F_{ug}(X_{ug}^i) \tag{4.8}$$

式中，$\boldsymbol{G}^i = \boldsymbol{B}_{ug}^i - \boldsymbol{A}_{ug}^i$。

未知向量 X_{ug}^i 包含了第 i 个谱单元的所有单元中所有 GLL 配置点上的位移和速度离散解。$\boldsymbol{G}^i$ 是第 i 个谱单元的全局线性矩阵。式（4.8）所示的线性代数方程组可以直接求解。

4.2　元模型混合自适应方法

元模型混合自适应方法（Hybrid and Adaptive Metamodeling Method，HAMM）[92、93]是为解决仿真优化中仿真高耗时及不同元模型的适应性问题而提出的一种应用混合代理模型进行优化的方法。此方法的流程如图 4.4 所示，其主要步骤如下。

（1）确定初始采样点。生成初始样本点，并调用耗时的目标函数和约束函数计算这些样本点的函数值。初始样本点的个数一般较少，由式 $m=(n+1)(n+2)/2$ 确定，式中的 n 是维数，m 是初始采样个数。这些点应用拉丁超立方采样方法获得，然后再加上 2^n 个设计空间端点，并计算其函数值来构造目标函数与约束函数的 Kriging、MARS 和 RBF 元模型。

（2）生成网格点。在原始设计空间内，每维上均匀取 k 个点，连接各维上的均分点形成 k^n 个正交网格点，用三个元模型计算所有网格点处的近似响应值。将三个元模型的函数值按升序排序，取出排在前面的 $k/100$ 个点，对应三个元模型作为 A、B、C 三组。

（3）对步骤（2）获得的三组再一次分组。根据元模型的函数值，二次分组为 $D=A\bigcap B\bigcap C, E=A\bigcap B, F=B\bigcap C, G=A\bigcap C$。

（4）确定新的采样点。依据分组中的网格点的数量确定权值，再由权值确定该分组中采样点的个数，然后采样，判断约束函数元模型是否满足，如果不满足转到步骤（2），增大 k 值；否则通过仿真获得函数值；如果其中最小值大于目前已获得的最小值，执行步骤（5），相反转到步骤（6）。

（5）局部优化。在已有采样点中取出三个函数值最小的点，以它们为中心构造新的子空间，然后应用已经构造的元模型在子空间内搜索，如果找不到更优点，则调整子空间。步骤：①网格化三个新的子空间，将每个子空间均匀网格化为 10^4 个网格；②计算三个元模型的函数值；③扩展子空间 5%，重复步骤（2），直到元模型的函数值不能再变小为止，重复次数≤3；④根据三个子空间中元模型的分析结果，分别选取 1 个点，共取 9 个点；⑤判断这 9 个点是否满足约束函数元模型，如果不满足，转到步骤（6），否则计算这 9 个点对应的仿真模型的值，并与现有的采样点合并，重新构造三个元模型；⑥重复①～⑤，直到满足要求为止，再转

到步骤（6）。

（6）重新构造混合元模型。通过选出的新采样点和现有的采样点重新构造元模型。

（7）反复迭代，直至收敛。

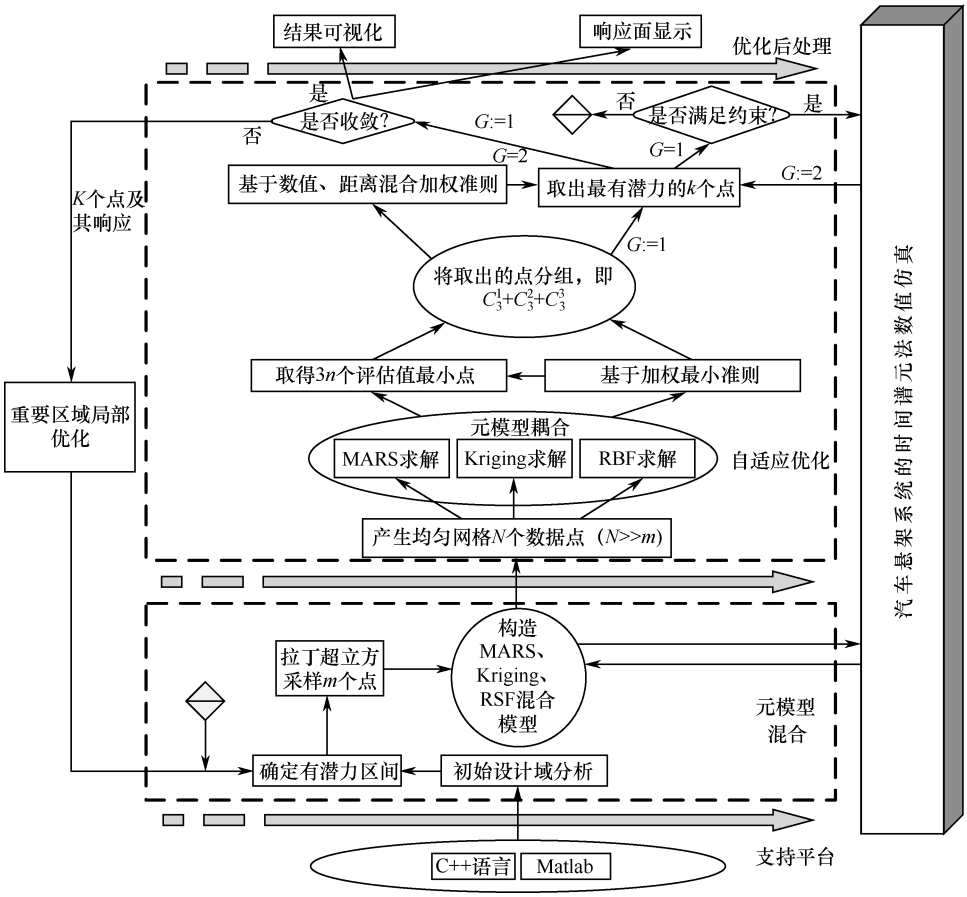


图 4.4　元模型混合自适应优化流程

4.3　处理与时间相关约束的方法

4.3.1　关键点集法（Critical Points Set Method，CPSM）

应用基于时间谱元法的序列二次规划法优化，处理依赖于时间的约束有两种方法[94]：一种是在 GLL 点上满足依赖于时间的约束；另一种是在单元的绝对极值点处满足依赖于时间的约束。第一种方法会离散数量较大的 GLL 点，使得 GLL

点相距很近，减小了在 GLL 点之间错过依赖于时间的约束的概率，因此会造成约束数量很大，并且 GLL 点多、插值点数高，计算资源的损耗很大；第二种方法不需要增加 GLL 点，只要满足仿真精度即可，只是对每个单元的 Lagrange 函数进行嵌套的一维优化，得到局部绝对极值点（由于每次迭代中的局部极值点都在变化，所以其计算量大）。文献[71]中说明了用 GLL 点法处理与时间相关的约束非常不稳定，且当单元数和插值点数达到一定规模时，才能找到最优值，否则找不到最优值；而采用关键点法时，当前迭代的局部最大值可能在下一次迭代不是局部最大值，这样收敛速度会变慢甚至不收敛。本章提出将所有单元的响应绝对最大值及其相邻的两个 GLL 点作为关键点集约束，对每一个谱单元的谱元点（即 GLL 点）进行 Lagrange 插值，将各单元组合获得模型，并采用 DIRECT 全局优化算法找到极大值点或极小值点，并与其周围最近的 GLL 点及谱单元的端点构成关键点集。如图 4.5 所示为处理与时间相关约束的关键点方法，对于时间段 $[t_2,t_3]$，其中的关键点集为 $\{t_2,t_{231},t_{23},t_{232},t_3\}$（$t_{23}$ 是谱单元 $[t_2,t_3]$ 中的极值点，t_{231}、t_{232} 分别为极值点左右两侧的 GLL 点，t_2、t_3 分别为谱单元的左右端点），将此关键点集的每一点的约束值作为结构动态响应优化的约束，这样不仅减少了约束的数量，而且不会因划分离散谱单元而漏掉时间约束。关键点集可以表示为

$$\Lambda=\left\{t_c,t_{\mathrm{GLL}_{1,2}},t_{e12}\right\} \tag{4.9}$$

式中，$t_c=\arg\min_t\left|L_{et}(t)\right|$；$t_{\mathrm{GLL}_{1,2}}$ 为点 t_c 两侧的 GLL 点；t_{e12} 为谱单元 e 的两个端点。

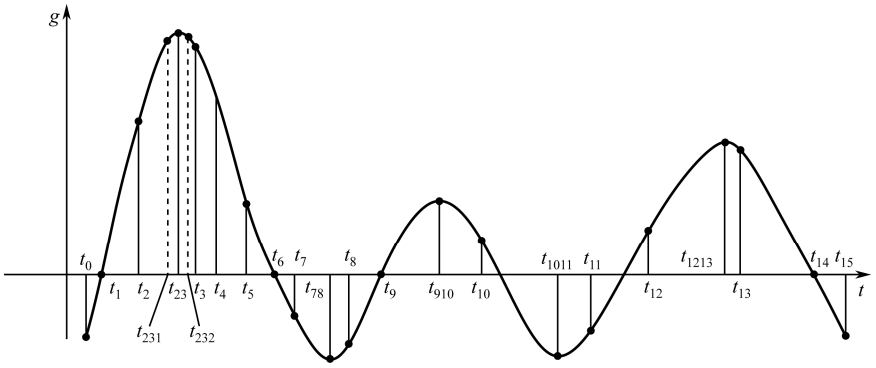


图 4.5　处理与时间相关约束的关键点方法

4.3.2　元模型混合自适应优化依赖于时间的约束

应用元模型混合自适应优化方法处理与时间相关的约束，是指通过谱元法求解动态响应，并计算与时间相关的约束函数，取出绝对值最大的约束函数值及其

对应的设计变量点，构造三个元模型，并用构造好的三个模型分别判断新的采样点，把不满足约束元模型的采样点删除[91]。

4.4 算例分析

汽车悬架系统可近似简化为图 4.6 所示的 5 自由度动力学模型。图中的 m_1 是驾驶员及其座位的质量，它由弹簧 k_1 和阻尼器 c_1 支撑在车体上。汽车车体的质量、前后车轴和车轮的质量分别为 m_2 、m_4 和 m_5。汽车车体由连于前后车轴的弹簧 k_2 和 k_3 及阻尼器 c_2 和 c_3 所支撑。k_4 、k_5 和 c_4 、c_5 表示轮胎的刚度和阻尼系数。车体对其质量中心的转动惯量用 I 表示。L 表示前后轮距长度。$f_1(t)$ 和 $f_2(t)$ 表示由于汽车行驶道路表面起伏不平所激起的前后轮的位移函数。$y_i(t)$ 是 5 个自由度的广义坐标。

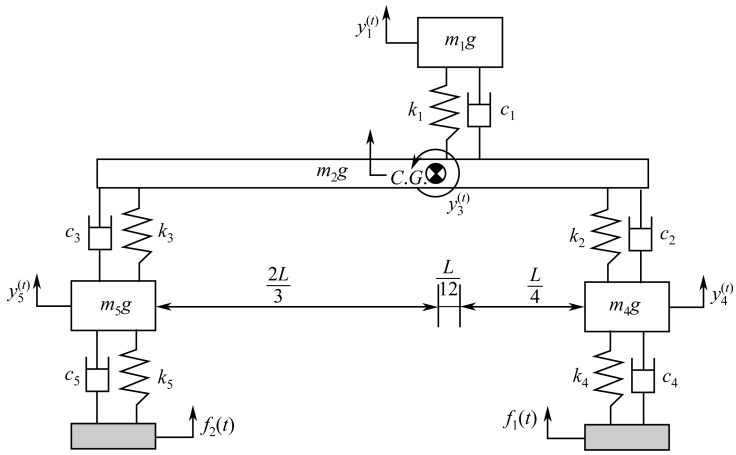


图 4.6　5 自由度动力学模型

4.4.1　汽车悬架系统动态响应优化设计问题一

路面轮廓如图 4.7 所示。汽车速度 $v=11.43\text{m/s}$。路面激励频率为 $\omega_i=\pi v/L_i$，$L_1=9.144\text{m}$，$L_2=3.6576\text{m}$，后轮到达前轮未知的时间间隔为 $t_\sigma=0.2667\text{s}$，$\omega_1=1.25\pi\ \text{rad/s}$，$\omega_2=3.125\pi\ \text{rad/s}$。

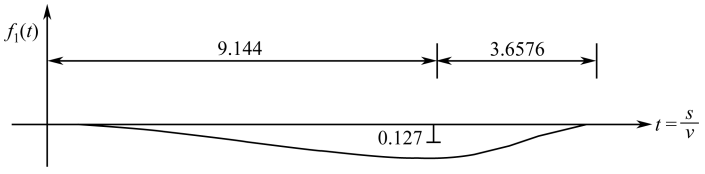


图 4.7　第一种路面轮廓

元模型混合自适应优化方法经过 54 次迭代后，算法收敛，其结果如表 4.1 所示，最优解的约束函数值如表 4.2 所示，优化迭代历程如图 4.8 所示。在时间谱元法中，离散了 100 个单元，在每个单元中进行了 6 次插值。

表 4.1　方法一与方法二的数值实验结果比较（问题一）

设计变量	初始值	最优值		
		方法一	方法二	文献[63]
$x_1 = k_1$	17512.60	10070.33	11258.24	8756.30
$x_2 = k_2$	52537.80	35563.50	35025.20	35025.20
$x_3 = k_3$	52537.80	83420.58	53880.14	42362.98
$x_4 = c_1$	1751.26	8627.98	8756.30	2257.37
$x_5 = c_2$	4378.15	9191.96	8750.48	13575.77
$x_6 = c_3$	4378.150	886.69	875.63	14010.08
d（目标函数）	8.448	5.821	5.841	6.538
优化迭代次数		54	60	
函数仿真次数		205	483	

注：表中（及其他表中）的方法一是指元模型混合自适应优化法，方法二是指关键点集法。

表 4.2　方法一与方法二的最优解约束函数值（问题一）

约　束	方法一	方法二
C1	–0.0358355047029415	–0.0367944773076181
C2	–0.000703142415041214	–1.11022302462516e – 16
C3	–0.0697869494510568	–0.0414955689609455
C4	–0.0180512292572415	–0.0188479468235111
C5	–0.0315933259038125	–0.0323011630470396

经过元模型混合自适应方法的优化，座位的最大加速度响应减小了 2.627m/s^2，且迭代 54 次，评估仿真模型仅仅 205 次；采用关键点集方法进行优化时，座位的最大加速度响应减小了 2.607m/s^2，迭代 60 次，评估仿真模型达 483 次，优化结果如表 4.1 所示，优化迭代过程如图 4.9 所示。两种方法约束都满足要求，如表 4.2 所示。

比较图 4.8 与图 4.9，可以看出元模型混合自适应方法能快速接近最优值点，迭代 12 次，目标函数为 5.839，和最优值 5.821 非常接近；而关键点集法需迭代 40 次才比较接近最优点。元模型混合自适应方法在第一次迭代时就获得了目标函数 6.215，而关键点集法在迭代 35 次时才能获得元模型混合自适应方法在第一次迭代的结果。

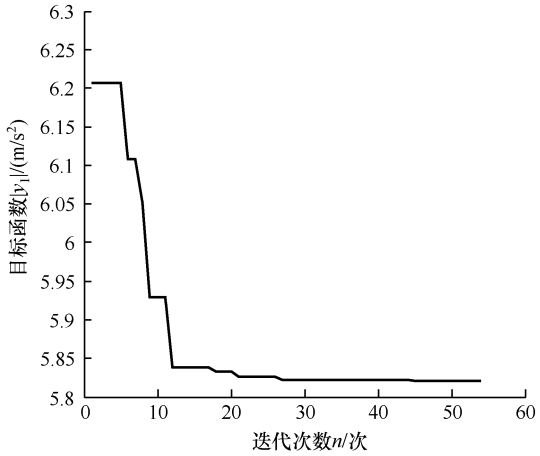


图 4.8　元模型混合自适应优化历程（问题一）

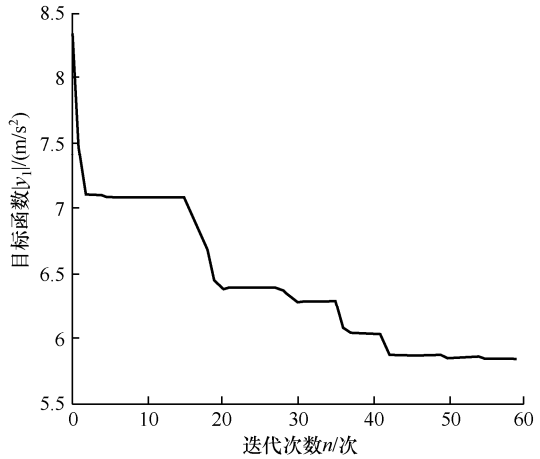


图 4.9　关键点集法的优化历程（问题一）

4.4.2　汽车悬架系统动态响应优化设计问题二

路面轮廓的位移函数 $f_1(t)$ 如图 4.10 所示。汽车速度 $v=24.384$ m/s 。$t_\sigma=0.125$ s 。两个环境参数为 ω_i=2π rad/s 和16π rad/s$(i=1,2,3,4)$ 。

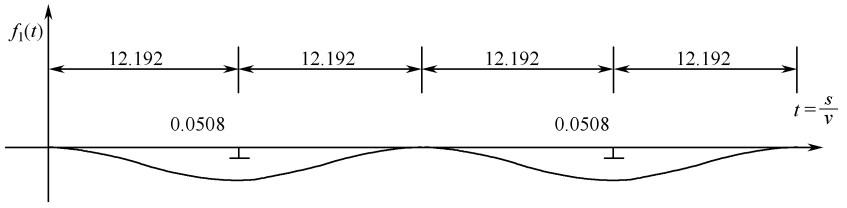


图 4.10　第二种路面轮廓

对于第二种路面轮廓（问题二），经过元模型混合自适应方法的优化，座位的最大加速度响应减小了 2.667m/s^2，且迭代 48 次，评估仿真模型仅仅 404 次，优化迭代过程如图 4.11 所示；采用关键点集法优化时，座位的最大加速度响应减小了 2.640m/s^2，迭代 96 次，评估仿真模型达 776 次。优化结果如表 4.3 所示，优化迭代过程如图 4.12 所示。两种方法约束都满足要求，如表 4.4 所示。在时间谱元法中，离散了 100 个单元，在每个单元中进行了 6 次插值。

表 4.3　方法一与方法二的数值实验结果比较（问题二）

设计变量	初始值	最优值		
		方法一	方法二	文献[63]
$x_1 = k_1$	17512.6	8197.86	8756.30	8756.30
$x_2 = k_2$	52537.8	35288.56	35025.20	35025.25
$x_3 = k_3$	52537.8	35427.53	35025.20	35025.25
$x_4 = c_1$	1751.26	9698.64	8756.30	1563.88
$x_5 = c_2$	4378.15	17612.97	14010.08	8041.79
$c_6 = c_3$	4378.15	13939.24	12469.35	6621.51
d（目标函数）	5.044	2.377	2.404	3.188
优化迭代次数		48	96	
函数仿真次数		404	776	

表 4.4　方法一与方法二的最优解约束函数值（问题二）

约　束	方法一	方法二
C1	–0.0453659323773259	–0.0447284701639262
C2	–0.0952961823885529	–0.0884401484273030
C3	–0.107986070112536	–0.106100647576319
C4	–0.0372927287523876	–0.0372203628095007
C5	–0.0439224637356453	–0.0438525043699045

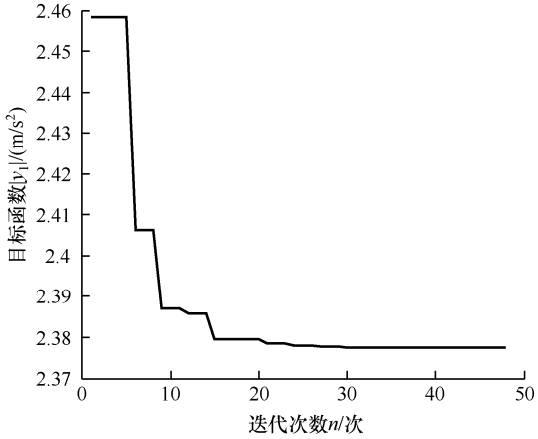


图 4.11　元模型混合自适应优化历程（问题二）

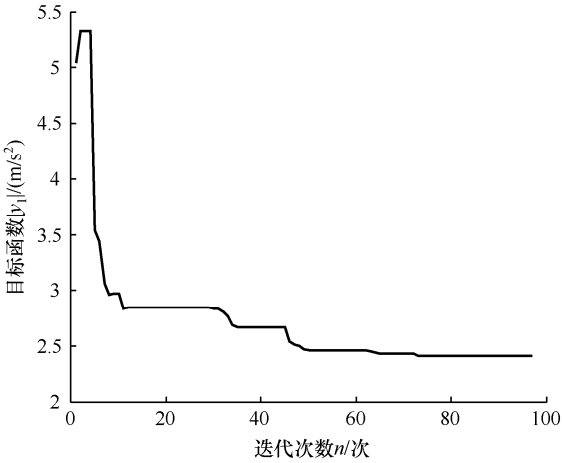


图 4.12　关键点集法的优化历程（问题二）

比较图 4.11 与图 4.12，可以看出元模型混合自适应方法迭代 9 次，目标函数为 2.387，和最优值非常接近；而关键点集法迭代 35 次，目标函数值为 2.667，只是比较接近最优点。元模型混合自适应方法在第一次迭代时就获得了目标函数 2.458，而关键点集法在迭代 50 次时目标函数值才为 2.456，才获得了元模型混合自适应方法在第一次迭代的结果。

从表 4.1 和表 4.3 中可以发现，问题一的 HAM 迭代 54 次，仿真次数仅为 205 次，而问题二的 HAM 迭代 48 次，仿真次数为 404 次，这是因为 HAM 每次迭代都要找到一些有潜力的点，然后用元模型测试这些点，对于前者，仿真次数少，迭代反而大，这在优化迭代过程的信息中可以明显看出来，每次迭代不满足约束的有潜力采样点较多而被删除；而后者每次迭代时不满足约束的有潜力采样点比较少。

当应用谱元法求解微分方程组时，文献[71]应用的 GLL 点法对谱单元数和插值次数非常敏感，本章提出的这两种方法对谱单元数和插值次数不敏感，因此都离散了 100 个单元，且在每个单元中进行了 6 次插值。

4.5　结论

本章应用谱元法将微分方程组转化为代数方程组，计算了整个时间内的响应；应用元模型混合自适应优化方法和关键点集法进行了优化。其中元模型混合自适应优化方法本身就是一个优化器，没有利用任何现有的优化器，而且比关键点集法更优越，即它不仅精确性高，迭代次数也非常少。例如，求解算例问题二，元模型混合自适应优化法仅仅迭代了 48 次，仿真 404 次就获得了比关键点集法更加

精确的结果，而关键点集法则需迭代 96 次，仿真 776 次。另外，元模型混合自适应优化法能快速找到最优解附近的点的特点，如求解算例问题一时仅仅迭代 12 次，目标函数为 5.839，和最优值 5.821 非常接近；求解算例问题二时仅仅迭代 9 次，目标函数为 2.387，和最优值 2.377 非常接近。

CPSM 克服了由于当前迭代的局部最大值可能在下一次迭代不是局部最大值，造成收敛优化结果不稳定甚至不收敛的缺点[71]，并且无论离散多少个谱单元和进行多少次插值都能获得满意的结果。

汽车悬架系统算例表明本章提出的两种方法均有效。但是不论是在精确性还是效率方面，HAM 更优于 CPSM。

4.6　本章小结

针对传统的结构动态响应优化方法的不足，本章提出了一种元模型混合自适应优化与时间谱元法相结合的方法，以及用于处理依赖于时间约束的关键点集方法。本章从 Bubnov-Galerkin 方法出发，深入探讨了在时间域内离散动态响应，将整体结构动力学方程转化成代数方程组，精确、高效地解出动态响应；依据优化问题的特点，采用自适应策略选择对应的元模型进行优化；在优化过程中利用均匀网格获取有潜力点的数量，并将局部优化与元模型混合自适应方法融合，使得优化结果更可靠。为了处理与时间相关的约束，本章提出将关键点及其相邻高斯-罗巴托-勒让德点组成集合关键点集的方法。对 5 自由度汽车悬架系统进行动态优化的结果表明，本章提到的两种方法都是有效的；但是不论是在精确性还是效率方面，元模型混合自适应与时间谱元法相结合的方法更优于关键点集方法。

第 5 章　基于 MARS 的动态响应优化算法

在 20 世纪 70 年代中期，Schmit[96、97]提出了响应面方法，在其后 30 多年的发展中证明它在大型结构非线性设计中是有效的和高效的。在文献中用得最多的响应面方法有 Kriging，径向基函数（Radial Basis Functions，RBF），多项式（Polynomial Response Surface，PRS），支持向量回归（Support Vector Regression，SVR）和多变量自适应回归样条（Multivariate Adaptive Regression Splines，MARS）等。其中，MARS 采用无参数回归技术，通过样条理论构造输入和输出之间的关系[88]，特别适合解决高维问题。本章利用 MARS 的特点，结合移动极限策略、置信域方法及数据驱动，研究了适合于基于计算昂贵的黑箱仿真模型的动态响应优化算法。

5.1　响应面方法

5.1.1　多项式

由 Box and Wilson 提出的多项式回归模型 PRS 是最常用的一种响应面模型[98]。对于 p 维问题，使用 m 次多项式对原函数 y 近似得到 $\hat{y}$：

$$\hat{y}=\beta_0+\sum_{i=1}^{p}\beta_i x_i+\sum_{i_1=1}^{p}\sum_{i_2=1}^{p}\beta_{i_1 i_2}x_{i_1}x_{i_2}+\cdots+\sum_{i_1=1}^{p}\sum_{i_2=1}^{p}\cdots\sum_{i_m=1}^{p}\beta_{i_1 i_2\cdots i_m}x_{i_1}x_{i_2}\cdots x_{i_m} \tag{5.1}$$

在实际应用中，通常使用二次多项式函数来近似原模型，即

$$\hat{y}=\beta_0+\sum_{i=1}^{p}\beta_i x_i+\sum_{i=1}^{p}\sum_{j=1\geqslant i}^{p}\beta_{ij}x_i x_j \tag{5.2}$$

将系数 β 按向量排序得到：

$$\hat{y}=\beta_0+\sum_{i=1}^{p}\beta_i x_i+\sum_{i=1}^{p}\sum_{j=1\geqslant i}^{p}\beta_{p+(i-1)\times(p-1)+j}x_i x_j \tag{5.3}$$

其待定系数 β 向量的元素个数为 $K+1$ 个：$K=2p+C_p^2$。

对 N 个采样点$(X_i, y_i)(i=1, 2, \cdots, N, N>K)$采用最小二乘法求待定系数，过程如下。

将(X_i, y_i)代入式（5.3），规整为

$$\boldsymbol{Xb}=\boldsymbol{y} \tag{5.4}$$

$$\boldsymbol{X}=\begin{bmatrix}1 & x_{11} & x_{12} & \cdots & x_{1K}\\ 1 & x_{21} & x_{22} & \cdots & x_{2K}\\ & \cdots & \cdots & \cdots & \cdots\\ 1 & x_{N1} & x_{N2} & \cdots & x_{NK}\end{bmatrix},\ \boldsymbol{b}=\begin{bmatrix}\beta_0\\ \beta_1\\ \cdots\\ \beta_K\end{bmatrix},\ \boldsymbol{y}=\begin{bmatrix}y_1\\ y_2\\ \cdots\\ y_N\end{bmatrix}$$

其中，$\boldsymbol{X}$ 矩阵的元素 x_{ij} 为：当 $1\leqslant j\leqslant p$ 时，x_{ij} 为一次项值，即第 i 个点的 j 维坐标值；当 $p+1\leqslant j\leqslant K$ 时，x_{ij} 为二次项值，为第 i 个点的 1 维坐标乘以 m 维坐标，满足 $p+(p-1)(l-1)+m=j, 1\leqslant m, 1\leqslant l, m\leqslant p$。

$\boldsymbol{b}$ 向量中的元素 β_j 与式（5.3）中一致。

由式（5.4）得到：

$$\boldsymbol{b}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y} \tag{5.5}$$

PRS 的效率高，适合低阶低维问题的近似，而且利用它可直接从标准化回归模型的系数识别出不同设计因素的特征[99]；其缺点是不适合高维非线性问题。PRS 还有多种变型[100]，包括双向 RSM、改进 RSM、随机 RSM 和自适应 RSM 等，可以改善简单的 PRS 的性能。

5.1.2　MARS

MARS 是 Jerome H. Friedman 于 1991 年提出的[101]。MARS 的目的是从大量的独立变量 $(x_1,\cdots,x_n)$ 中预测一个连续的输出变量 $\hat{f}$。MARS 具有综合样条回归（Smoothing Spline Regression）、累加回归（Additive Regression）、递归自分回归（Eecursive Partitioning Regression）的诸多优点，以“前向”和“后向”算法逐步筛选因子，与其他方法相比，它具有很强的自适应性，而且它对模型预测的精度也比较高，因而在实际工程中得到了广泛的应用。

在多元情况下，由于样本空间的扩张，使得怎样划分空间成为至关重要的问题，而 MARS 就是一种泛化能力很强的专门针对高维数据的回归方法。该回归方法以样条函数的张量积作为基函数，基函数的确定（张量积的变量的个数及变量的分割点）和基函数的个数都是自动完成的，不需要人工选定。

MARS 并不要求空间划分是不相交的，只要它们的并集能覆盖整个取值空间即可，每个小的划分区域对应一个系数，输入变量 x 落入区域对应的系数相加就是它的预测值，这样可以得到连续的函数估计，而且对于只有少数几个变量具有交互作用的情形，该方法具有更大的弹性。重要的是，该方法容易确定各个变量的累加贡献和多个不同变量的交互作用。

对于一个作为系统输入变量的自变量集 $x=(x_1,\cdots,x_p)$ 和一个作为系统输出变量的因变量集 $y=(y_1,\cdots,y_q)$，它们之间存在的关系如下：

$$y_j = f_j(x_1,\cdots,x_p)+\varepsilon_j \tag{5.6}$$

式中，f_j 是一个确定性函数（取 q 个值）；而 ε_j 是一组随机变量（q 个），反映了系统的随机扰动，按照惯例，ε_j 的期望值被设为零，即 $E(\varepsilon_j)=0$。

MARS 的目标是通过一系列训练数据 $\{y_{i_1},\cdots,y_{i_q},x_{i_1},\cdots,x_{i_p}\}_{i=1}^{N}$ 逼近得到一个可用的近似函数 $\hat{f}_j$ 来替代 f_j 对系统进行分析、计算。和其他逼近拟合方法相比，MARS 计算快速，可以处理数据量较大的样本，最后得到的近似函数具有很强的可解释性。MARS 算法可以准确地得出输入变量与输出变量之间的内在关系。

MARS 得到的最终近似模型为基函数集 $\{B_m(x_1,\cdots,x_p)\}_0^M$ 的展开形式：

$$\hat{f}(x_1,\cdots,x_p)=\sum_{m=0}^{M}a_mB_m(x_1,\cdots,x_p) \tag{5.7}$$

其中，系数 a_m 通过对训练数据的最小二乘拟合法计算：

$$\{a_m\}_0^M=\arg\min\sum_{n=1}^{N}\left[y_n-\sum_{m=0}^{M}a_mB_m(x_{n_1},\cdots,x_{n_p})\right]^2 \tag{5.8}$$

MARS 算法中基函数的形式为

$$B_m(x_1,\cdots,x_p)=\prod_{k=1}^{K_m}b_{k,m}(x_v(k,m)\,|\,P_{k,m}) \tag{5.9}$$

$B_m(x_1,\cdots,x_p)$ 由若干个 $b_{k,m}$ 相乘得到，$b_{k,m}$ 由输入变量 $x_{v(k,m)}$ 和参数集 $P_{k,m}$ 决定。

其中，$b_{k,m}$ 的形式如下：

$$b_{k,m}(x\,|\,s,t)=[s(x-t)]_+ \tag{5.10}$$

式中，下标表示正数部分，即 $[z]_+=\begin{cases}z & z>0\\ 0 & \text{其他}\end{cases}$。

式（5.10）中的参数为节点位置 $-\infty\leqslant t\leqslant+\infty$ 与截断方向 $s=\pm1$，则式（5.9）中的 $P_{k,m}=(s_{k,m},t_{k,m})$。

MARS 算法的目的就是通过一个前向迭代过程（Forward Iterative Procedure）和一个后向迭代过程（Backward Iterative Procedure）求出一个基函数集 $\{B_m(x_1,\cdots,x_p)\}_0^M$，使目标函数 $\hat{f}$ 的近似性满足要求。前向过程有意地对训练数据进行过拟合建模，因此得到的基函数个数也比较多；而后向过程则会选择性地删除部分基函数，使最终得到的模型具有最高的拟合度。

1. 前向迭代生成基函数算法

模型的初始基函数集（迭代次数 I=0）只包含一个常数项基函数，即

$$B_0(x)=1 \tag{5.11}$$

对于每一次迭代（$I>1$）都会生成两个新的基函数：

$$B_{2I-1}(x) = B_l(x)b(x_v \mid P) \tag{5.12a}$$

$$B_{2I}(x) = B_l(x)b(x_v \mid \overline{P}) \tag{5.12b}$$

式中，$B_l(x)$ 为先前迭代过程 $(0 \leqslant l \leqslant 2I-2)$ 中生成的基函数；x_v 为输入变量；$b(\cdot \mid P)$ 的形式如式（5.10）所示，其中

$$P = (s,t) \qquad \overline{P} = (-s,t) \tag{5.13}$$

式中，t 为基函数的节点位置。

在 MARS 算法中，每次迭代选择合适的节点对最后模型拟合的精度来说至关重要。为了在节约计算时间的情况下尽量增加模型精度，没有必要对每个点都进行测试，判断该点作为新增基函数的节点是否合适，因此应该对每个变量引入最小步长 L，从而在大量的数据中选取少量可能的节点进行计算。在不降低模型精度的情况下，确定相邻候选节点的间隔数据点个数 L 的计算法则如下：

$$L(a) = -\log_2\left[-\frac{1}{pN}\ln(1-a)\right]/2.5 \tag{5.14}$$

式中，$0.01 \leqslant a \leqslant 0.05$，这样可以合理缩小候选节点的选择范围，在要求精度几乎不变的情况下，把逼近速度提升 L 倍。

在式（5.12）中产生的新的基函数可以用 3 个参数 (l^*, v^*, P^*) 加以区分，即父基函数 $B_l(x)$，输入变量 x_v 与方程参数 P。通过对训练数据进行最小二乘拟合可以计算出以上 3 个参数：

$$(l^*, v^*, P^*) = \arg\min \sum_{n=1}^{N}[y_n - \sum_{i=0}^{2I-2} a_i B_i(x_n) - a_{2I-1}B_l(x_n)b(x_{vn} \mid P) - a_{2I}B_l(x_n)b(x_{vn} \mid \overline{P})]^2 \tag{5.15}$$

这些优化参数值 (l^*, v^*, P^*) 可以在第 I 次迭代中用于生成新的基函数。每次产生新基函数后，这些参数将被保存起来，作为潜在的父基函数供后继迭代遍历选择。前向过程中产生基函数的形式如式（5.9）所示，即通过一个单变量方程与一个已有的父基函数相乘得到一个新的基函数。整个迭代过程会一直继续下去，直到模型中的基函数个数达到用户定义的基函数最大个数或模型精度已满足要求。

2. 后向迭代选择最终模型

前向过程之所以故意对训练数据进行过拟合建模，是由于 MARS 算法只允许用先前生成的基函数来构建新的基函数，这样最初产生的那些基函数可能对最终模型的贡献较小或无贡献，这些基函数的作用仅仅是产生后继的基函数。因此，MARS 的前向迭代过程允许构造大量的基函数，也正因为如此，后向迭代过程对 MARS 算法显得尤为重要。

MARS 后向建模采用每次循环删除一个基函数得到子模型的方法，使用一般交叉实验（Generalized cross validation，GCV）准则：

$$\mathrm{GCV}(M)=\frac{1}{N}\sum_{i=1}^{N}[y_i-\hat{f}_M(x_i)]^2\Big/\left[1-\frac{C(M)}{N}\right]^2 \tag{5.16}$$

式中，$C(M)=\mathrm{trace}(\boldsymbol{B}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}})+1+\mathrm{d}M$; $\mathrm{trace}(\boldsymbol{B}(\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B})^{-1}\boldsymbol{B}^{\mathrm{T}})+1$ 为该模型有效系数的个数，通常与该模型所包含的基函数个数相同；d 为惩罚因子，在 2 到 4 之间，一般取值为 3。

可以通过 GCV 准则计算各个子模型的预测误差，在子模型集里取出一个预测误差最小的子模型作为最终输出的模型。

MARS 算法得到模型的最终形式为

$$\hat{f}(x)=a_0+\sum_{m=1}^{M}a_m\prod_{k=1}^{K_m}[s_{km}(x_v k,m)-t_{km}]_+ \tag{5.17}$$

式中，a_0 为常量基函数 B_1 的系数；式子后面的和为基函数 B_m；$s_{km}=\pm1$。

但是这种形式的模型解释性较差，为了使模型易于解读，可对式（5.17）进行 ANOVA 分解，使模型内部变量之间的交叉关系与基函数之间的叠加关系更加直观，分解后得到的模型可写为如下形式：

$$\hat{f}(x)=a_0+\sum_{K_m=1}f_i(x_i)+\sum_{K_m=2}f_{ij}(x_i,x_j)+\sum_{K_m=3}f_{ijk}(x_i,x_j,x_k)+\cdots \tag{5.18}$$

令 $V(m)=\{v(k,m)\}_1^{K_m}$ 为第 m 个基函数的变量集，则式（5.18）的第一项表示所有只含一个变量的基函数 $f_i(x_i)=\sum_{K_m=1,i\in V(m)}a_mB_m(x_i)$，第二项表示所有含两个变量的基函数 $f_{ij}(x_i,x_j)=\sum_{K_m=2,(i,j)\in V(m)}a_mB_m(x_i,x_j)$，第三项表示所有含三个变量的基函数 $f_{ijk}(x_i,x_j,x_k)=\sum_{K_m=3,(i,j,k)\in V(m)}a_mB_m(x_i,x_j,x_k)$，以此类推。

这样对模型的分析更加直观，可以清楚地分辨该模型是只含相加模型或哪些变量之间存在交叉；对于二维或三维问题，在图形界面上观察更加直观。

3. 模型后处理

选择模型后，基函数具有 $b(x\,|\,s,t)=[s(x-t)]_+$ 形式，若要使模型连续且具有一阶连续导数，应该把每个方程相应地替换为立方截断方程形式：

$$\begin{cases} C(x\,|\,s=-1,t_-,t,t_+)=\begin{cases}-(x-t) & x\leqslant t_-,\\ p_-(x-t_+)^2+r_-(x-t_+)^3 & t_-<x<t_+,\\ 0 & x\geqslant t_+\end{cases}\\ C(x/s=+1,t_-,t,t_+)=\begin{cases}0 & x\leqslant t_-,\\ p_+(x-t_-)^2+r_+(x-t_-)^3 & t_-<x<t_+,\\ x-t & x\geqslant t_+\end{cases}\end{cases} \tag{5.19}$$

式中，

$$p_+ = (2t_+ + t_- - 3t)/(t_+ - t_-)^2$$
$$r_+ = (2t - t_+ - t_-)/(t_+ - t_-)^3$$
$$p_- = (3t - 2t_- - t_+)/(t_- - t_+)^2$$
$$r_- = (t_- + t_+ - 2t)/(t_- - t_+)^3$$

$C(x|s,t_-,t,t_+)$ 连续且有一阶导数，它的二阶导数在 $x = t_\pm$ 处不连续，每一个节点 t 确定一个线性截断函数，而确定一个立方截断函数需要 t，t_+，t_-三个节点。

所有的 $b(x|s,t)$ 都被相应的 $C(x|s,t_-,t,t_+)$ 代替，中心节点 t 不变。

每一个基函数都有一个节点集 $\{t_{km}\}_1^{K_m}$，把所有的具有相同变量集 $\{v(k,m)\}_1^{K_m}$ 的基函数集中在一起，这样所对应的节点集可以看作 K_m 维空间上的一系列点集，这些点集投影到相应的 K_m 轴，即可得到相应变量的节点 t 的位置。t_+，t_-则分别为在同一轴上两个相邻的节点的中间位置。

模型后处理的好处是确保模型在变量空间内处处存在一阶导数，从而使模型更光滑，在计算量增加不大的情况下可使拟合精度稍微提升；而且区域之间的边界处更加平滑，当需要对模型进行三维图形显示时，立方截断方程会使模型输出更加光滑[101]。

5.2　基于 MARS 的优化技术

5.2.1　优化问题

一般的非线性约束优化问题描述为

$$\begin{aligned} \min_{x\in R^n} \quad & f_0(x) \\ \text{s.t.} \quad & f_j(x) = 0, (j = 1,\cdots,m_e) \\ & f_j(x) \leqslant 0, (j = m_e + 1,\cdots,m) \\ & A_i \leqslant x_i \leqslant B_i, (i = 1,2,\cdots,n) \end{aligned} \tag{5.20}$$

式中，$f_0(x)$ 为目标函数；$f_j(x)$ 为约束函数；A_i、B_i 分别为设计变量的上下限。

5.2.2　近似概念

当求解原问题时，在每一个子步骤中可用一个局部近似问题 $\hat{f}_j(x)$ 来代替原问题 $f_j(x)$。这里采用二次响应面（Quadratic Response Surface，QRS）和 MARS 这两种近似模型。原问题在某一步骤中可以表示如下：

$$\begin{aligned}\min_{x\in R^n}\quad & \overline{f}_0^{(k)}(x)\\ \text{s.t.}\quad & \overline{f}_j^{(k)}(x)=0,(j=1,\cdots,m_e)\\ & \overline{f}_j^{(k)}(x)\leqslant 0,(j=m_e+1,\cdots,m)\\ & A_i^{(k)}\leqslant x_i\leqslant B_i^{(k)},A_i\leqslant A_i^{(k)},B_i^{(k)}\leqslant B_i,(i=1,2,\cdots,n)\end{aligned}\tag{5.21}$$

式中，k 为当前迭代数；$A_i^{(k)}$、$B_i^{(k)}$ 为当前移动极限的下限和上限；$\overline{f}_j^{(k)}(x)$ 为原函数 $f_j^{(k)}(x)$ 的近似函数。

由于式（5.21）比较简单，所以应用标准优化算法即可求解。第 k+1 步迭代是从 k 步开始的。搜索子区间的大小和位置通过移动极限策略来确定。

5.2.3 基于 MARS 的动态优化策略

1. 移动极限策略（Move Limit Indicator Strategy，MLIS）

移动极限策略[102]（Move Limit Indicator Strategy）是由 Vassili Toropov 等于 1996 年提出来的，其基本思想是：求解近似优化子问题后，一个新的搜索子区域必须确定，即它的大小和位置都必须指定；它主要包括 7 个指标，这是该方法的核心，它们反映了近似的质量和优化收敛历史。

第一个指标是近似的功能质量指标。它是指最大的约束在设计过程中相对逼近的误差，表示为

$$E=\left|\frac{\overline{F}_j^{(k)}(x^{(k)})-F_p^{(k)}(x^{(k)})}{F_p^{(k)}(x^{(k)})}\right|\tag{5.22}$$

式中，$\overline{F}_j^{(k)}(x^{(k)})$ 为近似约束在子区间最优点的最大值；$F_p^{(k)}(x^{(k)})$ 为原约束在子区间最优点的最大值。

函数近似的质量分为“坏”，“合理”或“好”。如果

$$E^{(k)}>0.25S^{(k)},\quad S^{(k)}=\max\left(\frac{B_j^{(k)}-A_j^{(k)}}{B_j-A_j}\right)\tag{5.23}$$

则称为“坏”近似；如果

$$E^{(k)}<0.01S^{(k)}\tag{5.24}$$

则称为“好”近似；否则，称为“合理”近似。

第二个指标是局部最优点在当前搜索子空间的位置。如果当前移动极限没有被激活，则称解为“内部”，否则称解为“外部”。

第三和第四个指标是有关移动历史的。定义最后两次迭代之间的方向向量间的夹角 θ 为

$$\theta=\frac{x^{(k)}-x^{(k-1)}}{\|x^{(k)}-x^{(k-1)}\|}\cdot\frac{x^{(k-1)}-x^{(k-2)}}{\|x^{(k-1)}-x^{(k-2)}\|}\tag{5.25}$$

如果$\theta \leqslant 0$，表示移动“向后”；如果$\theta > 0$，表示移动“向前”。另外，当$\theta \leqslant \Theta$（Θ 是用户给定的系数，$0 < \Theta < 1$）时，收敛历史表示为“弯曲”，否则表示“垂直”。

第五个指标是收敛准则，指当前搜索子区间的大小，分为“小”或“大”。这和第一个指标有关，当“坏”近似且$S^{(k)} < 0.005$时，子区间称为“小”；当出现“合理”或“好”近似且$S^{(k)} < 0.1$时，子区间也称为“小”，否则称为“大”。

第六个指标是有关最活跃的约束的，分为“近”或“远”。如果约束的最大值在某一范围，即$\max(F_j^{(k)}(x^{(k)})) \in [0.9,1.1]$，称为“近”，否则称为“远”。

缩减和扩大搜索子空间，采用

$$B_j^{(k+1)} - A_j^{(k+1)} = \frac{1}{\tau}(B_j^{(k)} - A_j^{(k)}) \tag{5.26}$$

式中，τ 为修改系数。

移动极限如表 5.1 所示。

表 5.1　移动极限

<table>
<tr><td colspan="7">近似为“坏”</td></tr>
<tr><td>“小”</td><td colspan="6">“大”</td></tr>
<tr><td rowspan="2">停止，没有收敛</td><td colspan="3">“垂直”</td><td colspan="3">“弯曲”</td></tr>
<tr><td colspan="3">适当减小或稍微扩大</td><td colspan="3">减小</td></tr>
<tr><td colspan="7">近似为“合理”</td></tr>
<tr><td colspan="3">“内部”</td><td colspan="4">“外部”</td></tr>
<tr><td>“小”</td><td colspan="2">“大”</td><td colspan="2">“向后”</td><td colspan="2">“向前”</td></tr>
<tr><td rowspan="2">停止，没有收敛</td><td>“近”</td><td>“远”</td><td rowspan="2">减小</td><td colspan="2">“垂直”</td><td>“弯曲”</td></tr>
<tr><td>减小</td><td>减小</td><td colspan="2">扩大</td><td>保持</td></tr>
<tr><td colspan="7">近似为“好”</td></tr>
<tr><td colspan="3">“内部”</td><td colspan="4">“边界”</td></tr>
<tr><td>“小”</td><td colspan="2">“大”</td><td>“向后”</td><td colspan="3">“向前”</td></tr>
<tr><td rowspan="2">停止，没有收敛</td><td>“近”</td><td>“远”</td><td rowspan="2">减小</td><td>“垂直”</td><td colspan="2">“弯曲”</td></tr>
<tr><td>减小</td><td>减小</td><td>保持模型并扩大区域</td><td colspan="2">保持模型并保持区域尺寸</td></tr>
</table>

2. MARS 和 MLS 结合

MARS 和 MLS 结合的步骤：首先通过实验设计获得一些设计点，然后在全局构造 MARS 响应面，确定搜索子区间的位置和大小；最后通过 SQP 优化算法获得近似问题的最优点。

在文献中，常用的实验设计策略有全因子和部分因子设计、中心复合设计（Central Composite Designs，CCD）[103]、田口[104]、D 最优设计[105]和拉丁超立方设计（Latin Hypercube Designs，LHD）[106、107]。其中，LHD 是由 McKay 等于 1979

年专门为仿真试验提出的一种计算机试验设计方法。它是一种充满空间的设计，使输入组合相对均匀地填满整个试验区间，并且每个试验变量水平只使用一次。如果一个设计点不影响响应指标且从试验变量设置中删除，试验设计仍然是没有任何点是重合的，这就是拉丁超立方设计的优势。在西方世界里，拉丁超立方设计广泛应用在仿真试验中。传统的试验设计抽样，由经验发现往往存在堆积点的问题[108]，即试验组合没有布满整个空间，所得模型也将不能代表整个参数变化区域。因此，在构造 MARS 模型时更需要试验设计填满空间。

图 5.1 详细说明了全局响应面 MARS 和 MLS 相结合的过程。首先，给定设计变量、目标函数、约束函数和设计空间，通过 LHD 获得设计点，这些点的函数

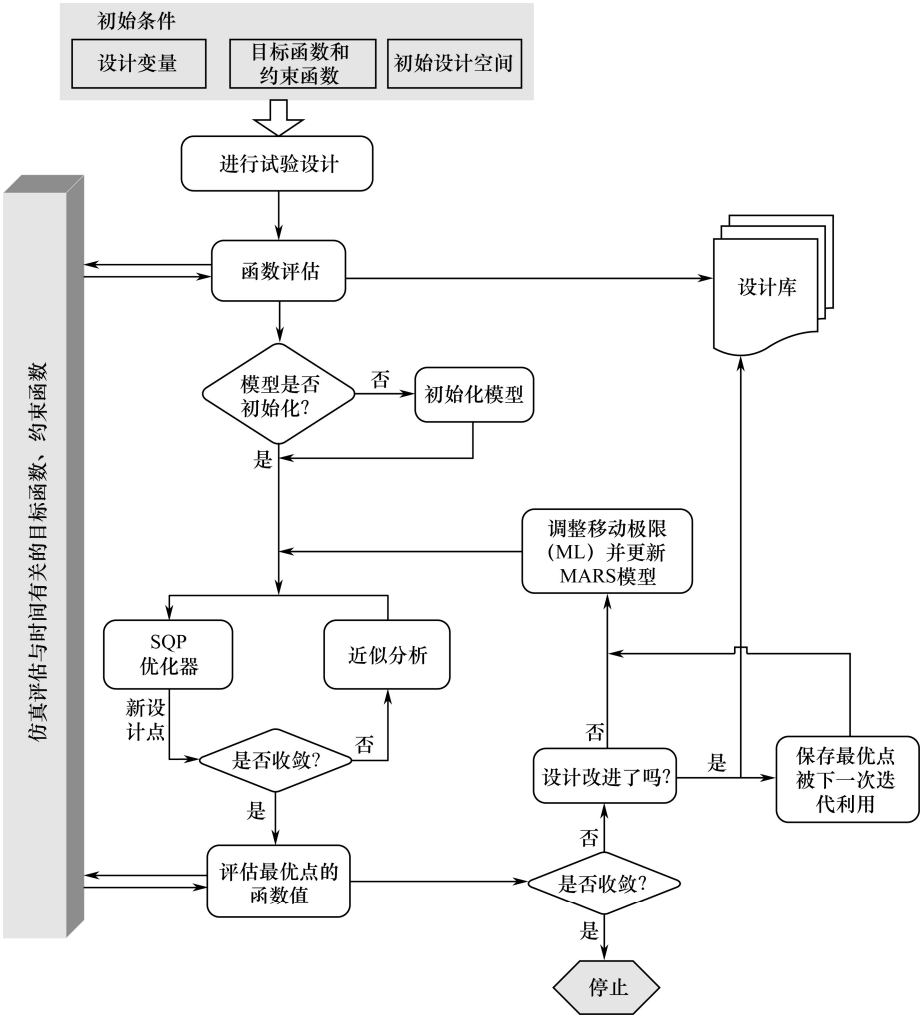


图 5.1　全局响应面 MARS 与 MLS 相结合的过程

值要通过计算昂贵的仿真评估得到。其次，构造 MARS 模型，调用 SQP 优化器寻找最优点，如果近似问题不收敛，就在当前的子区域内 LHD 采样，重新构造全局 MARS 响应面，再调用 SQP 寻优，直到子问题收敛为止，并通过计算昂贵的仿真评估获得最优点的函数值；如果原问题收敛，则停止迭代，否则和上一次迭代结果相比较，看是否有改进，如果有改进，则将当前的值保存在数据库中，被下一次迭代利用；如果没有改进，则直接调用 MLS 确定下一次迭代的子区间并采样，然后仿真，再更新模型，重新进入优化步，这样反复迭代直到原问题收敛。

当相邻两次最优值的相对误差小于一个给定的常数，或相邻两次最优值点的设计变量取值之差小于一个给定的常数时，优化过程终止。具体终止条件如下：

$$\begin{cases} \left| x^{(k+1)} - x^{(k)} \right| \leqslant \varepsilon_1 \\ \dfrac{\left| f(x^{(k+1)}) - f(x^{(k)}) \right|}{f(x^{(k)})} \leqslant \varepsilon_2 \end{cases} \tag{5.27}$$

式中，$\varepsilon_1, \varepsilon_2$ 为由设计者给定的两个很小的正常数。

3. 信任域方法

最早的信任域方法（Trust Region Method，TRM）是由 Levenberg 在 1944 年提出来的，其基本思想是：在局部可信赖的区域内，构造近似模型，解局部优化问题，解得结果有两种情况，一种是可信，即 $f(x_{k+1}) < f(x_k)$，另一种是不可信，即 $f(x_{k+1}) > f(x_k)$；当可信时，增大信任域半径，再构造局部近似模型，解局部优化问题；当不可信时，减小信任域半径，再构造局部近似模型，解局部优化问题；重复上述过程直至收敛。

下面给出一个简单的例子来阐明信任域法的基本思想。首先考虑两个变量的无约束优化问题：$f(X) = -10x_1^2 + 10x_2^2 + 4\sin(x_1 x_2) - x_1^2 x_2^2 + x_1^4 + x_2^4$。

然后从（0.7, −3.3）开始研究。如图 5.2 所示，当前的点在图中是黑圆点，k 是迭代次数。

围绕 x_k 构造一个局部近似响应面，如二次响应面 $Q(x)$。如图 5.3 所示，黑色圆表示二次响应面。围绕 x_k 周围定义一个信任域，信任域半径为 Δ_k。这样在信任域内搜索最小值，最小值点在十字处 x_{k+1}。

如果 $f(x_{k+1}) < f(x_k)$，则 $Q_k(x)$ 是 $f(x)$ 很好的局部近似，这就等于给了我们一个忠告。接着将当前的点移动到 x_{k+1}，准备进行下一次迭代。既然 $Q_k(x)$ 在局部这样好地近似原函数，则下一次迭代将增加信任域半径 Δ_k。在当前点及增加半径的新信任域内重新构造响应面 $Q_{k+1}(x)$，然后再搜索最小值点。最小值点在十字处 x_{k+2}，如图 5.4 所示。

如果 $f(x_{k+2}) < f(x_{k+1})$，则继续增加信任域半径 Δ_k。移到新点 x_{k+2}，重新构造响应面。重复上述过程，如图 5.5 所示。

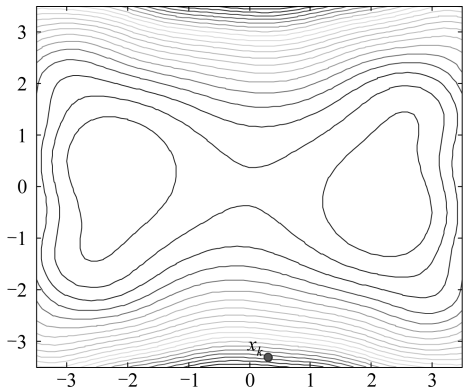


图 5.2　开始点（0.7, −3.3）

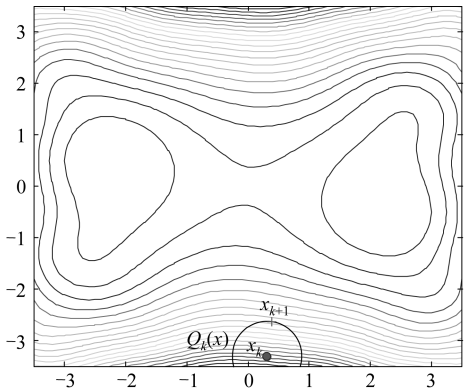


图 5.3　在点（0.7, −3.3）处构造二次响应面

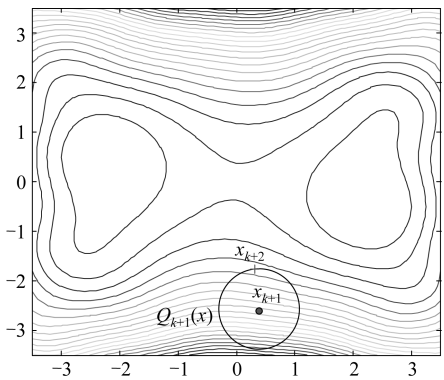


图 5.4　第 1 次迭代并构造二次响应面

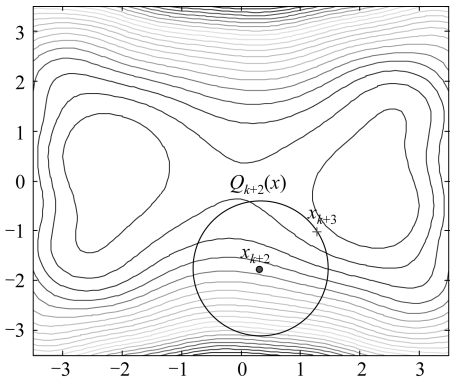


图 5.5　第 2 次迭代并构造二次响应面

如果某次迭代的 $f(x_{n+1}) > f(x_n)$，则 $Q_n(x)$ 不能近似原函数 $f(x)$。这样在下一次迭代时需减小信任域半径 Δ_k，重新构造响应面 $Q_{n+1}(x)$，优化得到最优点 x_{n+2}，如图 5.6 和图 5.7 所示。

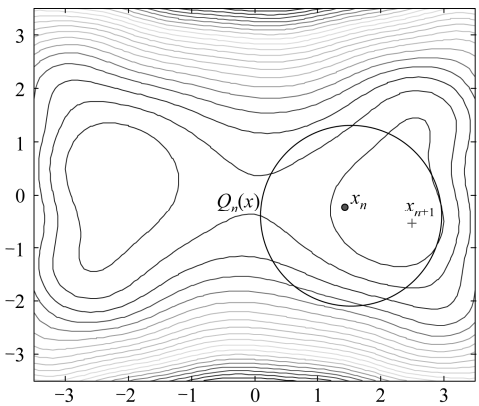


图 5.6　第 3 次迭代并构造二次响应面

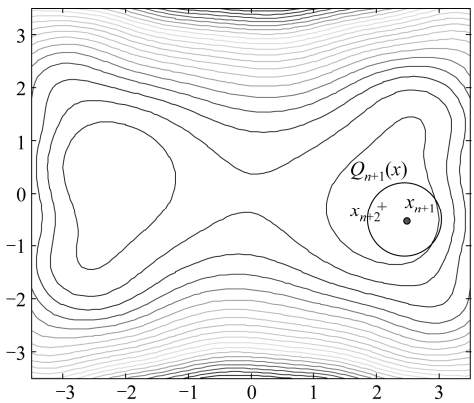


图 5.7　第 4 次迭代并构造二次响应面

4. 数据驱动

数据库是按照数据结构来组织、存储和管理数据的仓库。在经济管理的日常工作中，常常需要把某些相关的数据放进这样的仓库，并根据管理的需要进行相应的处理。在财务管理、仓库管理、生产管理中也建有众多的数据库，使其可以利用计算机实现财务、仓库、生产的自动化管理。然而，在这里我们要管理的不是财务、商品，而是计算机实验。计算机实验管理是序列近似优化的一个非常重要的方面。数据库是存储、检索和管理实验数据的一个有效方法。因此，将数据库技术引入序列近似优化中是非常好的创意。在优化过程中，当需要函数值或约束函数值时，首先需要检索数据库，确定它们是否已经存在于数据库中，如果存在，则直接取出即可，这样就避免了十分耗时的计算昂贵的分析仿真。

本章的数据驱动技术是在每一次迭代中找到一些替代点，来减少仿真次数。对于每一个采样设计点，都要检索数据库，找出 Δ/n 内的点（Δ 是置信域半径，n 是采样点数），也就是说，对于当前的采样点，如果在其 Δ/n 的范围内数据库中存在，则用数据库中的点来代替该采样点，这样可节省大量的计算昂贵的仿真分析；如果不存在，只能进行计算昂贵的仿真分析，然后将结果存入数据库中。这样，随着迭代的进行，数据库越来越成熟。因此，在优化过程中，只有在数据库中不存在的点才需要计算昂贵的仿真分析。

5. 增广拉格朗日方法

一般增广拉格朗日函数 Φ 可表示为

$$\Phi(x,\lambda,r_\rho)=f(x)+\sum_{j=1}^{m_e}(\lambda_j h_j(x)+r_\rho h_j(x)^2)+\sum_{j=m_e+1}^{m}(\lambda_{p+j}\Psi_j(x)+r_\rho\Psi_j(x)^2) \tag{5.28}$$

式中，λ_i 为拉格朗日乘子向量；r_ρ 为惩罚系数；$\Psi(x)$ 通过

$$\Psi_j(x)=\min\left\{g_j(x),-\frac{\lambda_{p+j}}{2r_\rho}\right\} \tag{5.29}$$

来定义。

当用增广拉格朗日方法解优化问题时，需要考虑两个重要因素：第一个是拉格朗日乘子的更新；第二个是惩罚系数的更新。由 Hestenes and Powell 提出的等式约束的更新方案为

$$\lambda_i^{k+1}=\lambda_i^k+2r_\rho h_i(x^k) \tag{5.30}$$

不等式约束的更新方案为

$$\lambda_{p+j}^{k+1}=\lambda_{p+j}^k+2r_\rho\min\left\{g_j(x^k),-\frac{\lambda_{p+j}^k}{2r_\rho}\right\} \tag{5.31}$$

6. 基于 DD 和 MARS 的优化技术

1）信任域模型管理

尽管通过建立序列增广拉格朗日响应面，优化能收敛到最小值点，但是在优化过程中如何评价增广拉格朗日响应面的近似质量呢？定义一个区域，近似响应面在其中是可信赖的，这就是信任域方法。

信任域测试评估当前的增广拉格朗日近似响应面 $\tilde{\Phi}(x^k,\lambda^k,r_\rho^k)$ 是否有效。定义信任域比 ρ^s 为

$$\rho^s=\frac{\Phi(x^k,\lambda^k,r_\rho^k)-\Phi(x^{k,s},\lambda^k,r_\rho^k)}{\tilde{\Phi}(x^k,\lambda^k,r_\rho^k)-\tilde{\Phi}(x^{k,s},\lambda^k,r_\rho^k)} \tag{5.32}$$

式中，s 为给定一组固定的 λ^k 和一个 r_ρ^k 的循环迭代，也称为内循环；k 为增广拉格朗日近似循环，也称为外循环；$\tilde{\Phi}(x^k,\lambda^k,r_\rho^k)$ 为近似增广拉格朗日函数；$\Phi(x^k,\lambda^k,r_\rho^k)$ 为真实增广拉格朗日函数。

如果 $\rho^s\leqslant 0$，表示近似质量差，减小信任域半径 Δ^s；如果 $\rho^s\in(0,1)$，则信任域半径可能不变、减少或增加。

$$\begin{aligned}&\Delta^{s+1}=\gamma_0\Delta^s,\ \rho^s\leqslant 0.25\\&\Delta^{s+1}=\Delta^s,\ 0.25<\rho^s<0.75\\&\Delta^{s+1}=\gamma_2\Delta^s,\ \rho^s\geqslant 0.75\end{aligned} \tag{5.33}$$

式中，$\gamma_0=0.25$；$\gamma_2=2$，如果 $\|x^{k,s}-x^k\|=\Delta^s$；$\gamma_2=1$，如果 $\|x^{k,s}-x^k\|<\Delta^s$。

如果 $\rho^s\leqslant 0$，本次迭代的结果被抛弃，按照式（5.33）减小信任域半径。

图 5.8 描述了全局 MARS 响应面信任域优化算法的流程。两个深颜色块表示拉格朗日更新，移动极限的调整和模型更新[109]。图 5.9 给出了详细流程。下面对数据驱动的序列近似优化（DDSAO）进行详细描述。

2）算法描述

步骤 0．初始化

给定初始设计向量 x^0，拉格朗日乘子向量 λ^0 和惩罚项 r_ρ 及 $\varepsilon_1,\varepsilon_2,\rho^0,\eta^0,\mu^0$，并且设 $k=0, s=0$。评估 $\Phi(x^k,\lambda^k,r_\rho^k)$ 和 $\nabla\Phi(x^k,\lambda^k,r_\rho^k)$。转到步骤 8。

步骤 1．系统评估和全局灵敏度分析

评估 $\Phi(x^{k,s},\lambda^k,r_\rho^k)$ 和 $\nabla\Phi(x^{k,s},\lambda^k,r_\rho^k)$。

步骤 2．计算置信域半径并测试

计算置信域半径 ρ^s。如果 $\rho^s>0.25$，按照式（5.33）更新置信域半径，设 $x^k:=x^{k,s}, s:=s+1$，并转到步骤 3，否则转到步骤 7。

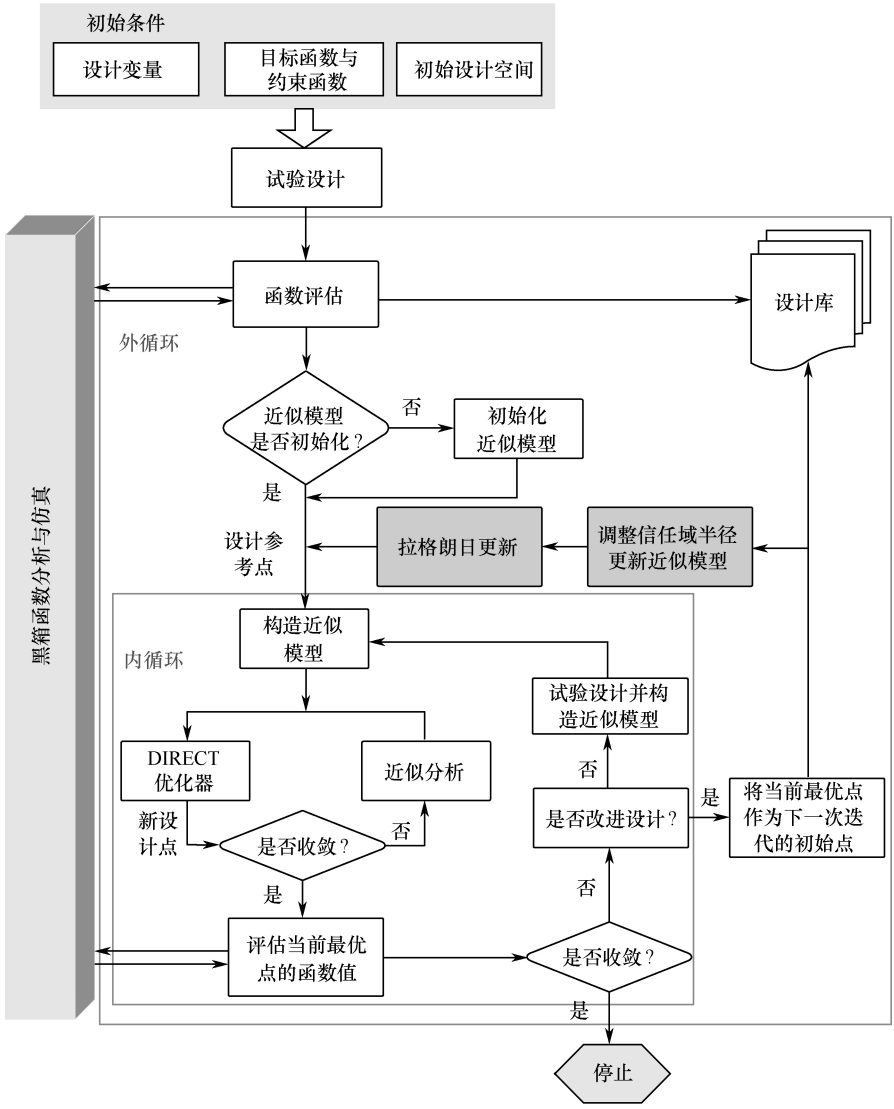


图 5.8　全局 MARS 响应面信任域优化算法的流程

步骤 3．检查增广拉格朗日函数梯度是否减少

计算 $P(\nabla\Phi(x^{k,s},\lambda^k,r_\rho^k))$，如果 $\|P(\nabla\Phi(x^{k,s},\lambda^k,r_\rho^k))\|\leqslant\mu^k$，然后转到步骤 4，否则转到步骤 8。

步骤 4．收敛检查

如果 $\|P(\nabla\Phi(x^{k,s},\lambda^k,r_\rho^k))\|\leqslant\varepsilon_1$，并且 $\|c(x^k)\|_\infty\leqslant\varepsilon^2$，则停止；如果 $\|c(x^k)\|_\infty\leqslant\eta^k$，则转到步骤 5，否则转到步骤 6。

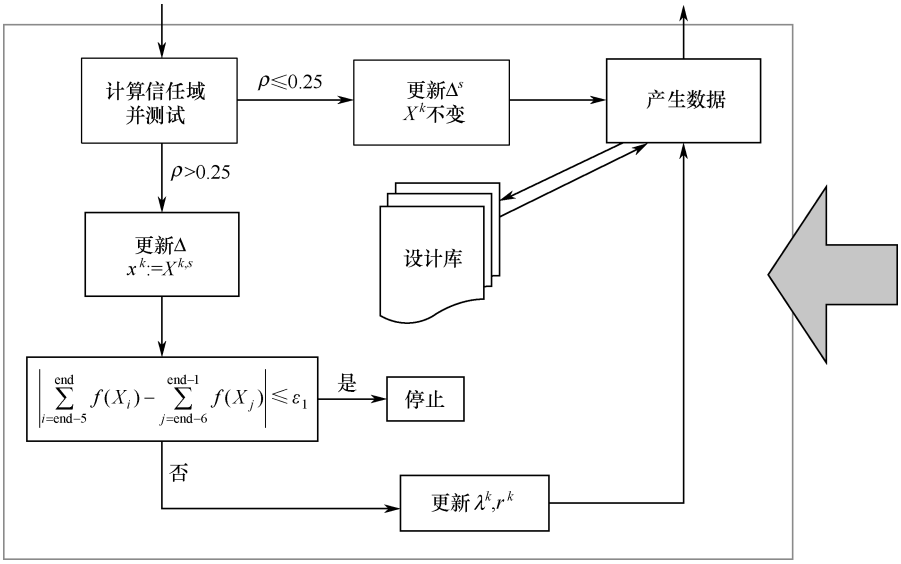


图 5.9　更新拉格朗日乘子和惩罚系数

步骤 5．更新拉格朗日乘子

按照式（5.30）和式（5.31）更新拉格朗日乘子，并且设 $r_\rho^{k+1} := r_\rho^k$，$\eta^{k+1} := \eta^k[\min\{1/r_\rho^{k+1}, \gamma_1\}]^{0.9}, \mu^{k+1} := \mu^k[\min\{1/r_\rho^{k+1}, 0.1\}], k := k+1, x^k := x^{k-1}, s := 0$。重新开始信任域半径设置，并转到步骤 8。

步骤 6．增加惩罚系数

设置 $\lambda^{k+1} := \lambda^k, r_\rho^{k+1} := 10 r_\rho^k, \eta^{k+1} := \eta^0[\min\{1/r_\rho^{k+1}, \gamma_1\}]^{0.1}$，$\mu^{k+1} := \mu^0[\min\{1/r_\rho^{k+1}, 0.1\}]$，$k := k+1, x^k := x^{k-1}, s := 0$。重新开始信任域半径设置，并转到步骤 8。

步骤 7．收缩移动极限

按照式（5.33）更新信任域半径，设 $s := s+1$，转到步骤 8。

步骤 8．产生数据库

在当前迭代点，应用某种采样技术获得设计数据。

步骤 9．构造近似增广拉格朗日

围绕点 (x^k, λ^k) 构造 MARS 增广拉格朗日响应面 $\tilde{\Phi}(x^k, \lambda^k, r_\rho^k)$。

步骤 10．解近似增广拉格朗日极小化问题

$$\tilde{\Phi}(x^{k,s}, \lambda^k, r_\rho^k) = \min_{x \in X^k} \tilde{\Phi}(x, \lambda^k, r_\rho^k), x^{k,s} \in X = \left\{ x \mid x \in S, \| x - x^k \| \leqslant \Delta^s \right\}$$

转到步骤 1。

5.3　实例分析

5.3.1　Hesse 函数

Hesse 函数（HE）有 6 个设计变量，18 个局部最小值，最小值为−310，目标函数和约束函数如下：

$$\begin{aligned}
&\min \quad -f(x)=25(x_1-2)^2+(x_2-2)^2+(x_3-1)^2+(x_4-4)^2\\
&\text{s.t.} \quad 2\leqslant x_1+x_2\leqslant 6\\
&\qquad x_1-3x_2\leqslant 2\\
&\qquad 4\leqslant (x_3-3)^2+x_4\\
&\qquad 4\leqslant (x_5-3)^2+x_6+(x_5-1)^2+(x_6-4)^2\\
&\qquad x_{1,2}\geqslant 0,1\leqslant x_3\leqslant 5,0\leqslant x_4\leqslant 6,1\leqslant x_5\leqslant 5,0\leqslant x_6\leqslant 10
\end{aligned} \tag{5.34}$$

Hesse 函数优化目标函数和移动极限的迭代历史如图 5.10 所示。

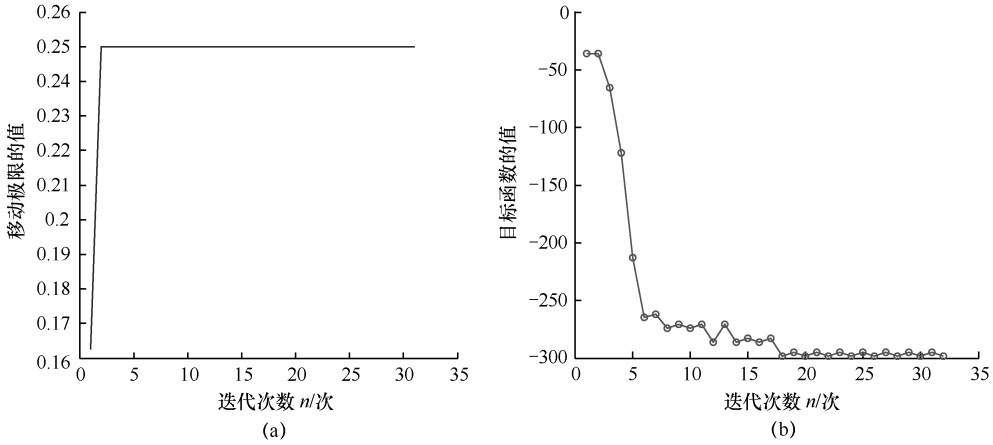


图 5.10　Hesse 函数优化目标函数和移动极限的迭代历史

Hesse 函数（HE）与 GRS/MARS 方法一起得到一个最优值点 $x^*=(5,1,5,6,5,10)$，目标函数值 $f(x^*)=-298.0$。

5.3.2　阶梯悬臂梁截面设计

如图 5.11 所示，悬臂梁自由端受一力 $P[P=50000(N)]$，其方向为竖直向下，杨氏模量 E=2.0e+8（N/cm^2），应力极限σ_b=14000.0（N/cm^2），弯曲极限 $y_b=0.5$cm，

梁长 L = 500cm，五段不同截面等长度，等于 100cm，即 $L_1 = L_2 = L_3 = L_4 = L_5 =$ 100cm，截面是长方形，高和宽表示为 H_i 、$W_i(i = 1,2,3,4,5)$。优化目标函数为梁的总质量最小，约束为梁自由端的变形小于等于 0.5cm，每一段截面的高与宽的比小于等于 20，每一部分的应力小于等于 σ_b。

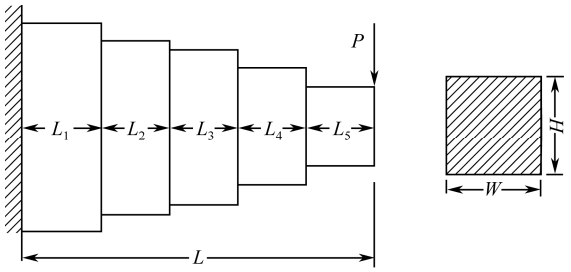


图 5.11　阶梯悬臂梁设计

悬臂梁问题描述为优化问题：

$$
\begin{aligned}
&\text{Minimize} \quad f = 100(x_1x_6 + x_2x_7 + x_3x_8 + x_4x_9 + x_5x_{10}) \\
&\text{Subject to} \quad g_1 = 0.5m_1x_1 / I_1 \leqslant 14000 \\
&\qquad\qquad\quad g_2 = 0.5m_2x_2 / I_2 \leqslant 14000 \\
&\qquad\qquad\quad g_3 = 0.5m_3x_3 / I_3 \leqslant 14000 \\
&\qquad\qquad\quad g_4 = 0.5m_4x_4 / I_4 \leqslant 14000 \\
&\qquad\qquad\quad g_5 = 0.5m_5x_5 / I_5 \leqslant 14000 \\
&\qquad\qquad\quad g_6 = y_5 \leqslant 0.5 \\
&\qquad\qquad\quad g_7 = x_1 / x_6 \leqslant 20 \\
&\qquad\qquad\quad g_8 = x_2 / x_7 \leqslant 20 \\
&\qquad\qquad\quad g_9 = x_3 / x_8 \leqslant 20 \\
&\qquad\qquad\quad g_{10} = x_4 / x_9 \leqslant 20 \\
&\qquad\qquad\quad g_{11} = x_5 / x_{10} \leqslant 20
\end{aligned}
\tag{5.35}
$$

其中，

$$
\begin{aligned}
&m_1 = 500p \quad I_1 = x_6x_1^3 / 12 \quad z_1 = p \times 4.5\text{e}4 / (EI_1) \\
&m_2 = 400p \quad I_2 = x_7x_2^3 / 12 \quad z_2 = p \times 3.5\text{e}4 / (EI_1) + z_1 \\
&m_3 = 300p \text{ ，} \quad I_3 = x_8x_3^3 / 12 \text{ ，} \quad z_3 = p \times 2.5\text{e}4 / (EI_1) + z_2 \\
&m_4 = 200p \quad I_4 = x_9x_4^3 / 12 \quad z_4 = p \times 1.5\text{e}4 / (EI_1) + z_3 \\
&m_5 = 100p \quad I_5 = x_{10}x_5^3 / 12 \quad z_5 = p \times 0.5\text{e}4 / (EI_1) + z_4
\end{aligned}
$$

$$
\begin{aligned}
y_1 &= p\times 2.333333\text{e}6/(EI_1)\\
y_2 &= p\times 1.333333\text{e}6/(EI_2)+100z_1+y_1\\
y_3 &= p\times 2.333333\text{e}6/(EI_3)+100z_2+y_2\\
y_4 &= p\times 1.333333\text{e}6/(EI_4)+100z_3+y_3\\
y_5 &= p\times 0.333333\text{e}6/(EI_5)+100z_4+y_4
\end{aligned}
\quad
\begin{aligned}
&5\leqslant x_i\leqslant 100, i=1,2,\cdots,6\\
&1\leqslant x_i\leqslant 10, i=7,8,\cdots,10
\end{aligned}
$$

对于悬臂梁设计问题，用 GRS/MARS 方法找到全局最优点为 $x^*=(59.83, 55.55, 50.48, 44.11, 33.31, 299, 278, 252, 220, 187)$，目标函数 $f(x^*)=62051.7485$。而商用优化软件 i-SIGHT 优化的结果为 $x^*=(59.84, 55.54, 50.48, 33.33, 299, 278, 252, 221, 192)$，目标函数 $f(x^*)=62311.94$。

悬臂梁目标函数和移动极限的迭代历史如图 5.12 所示。应用 DDSAO 方法求解阶梯悬臂梁截面设计问题，初始点选在设计域中心。求解结果如图 5.13 和表 5.2 所示。

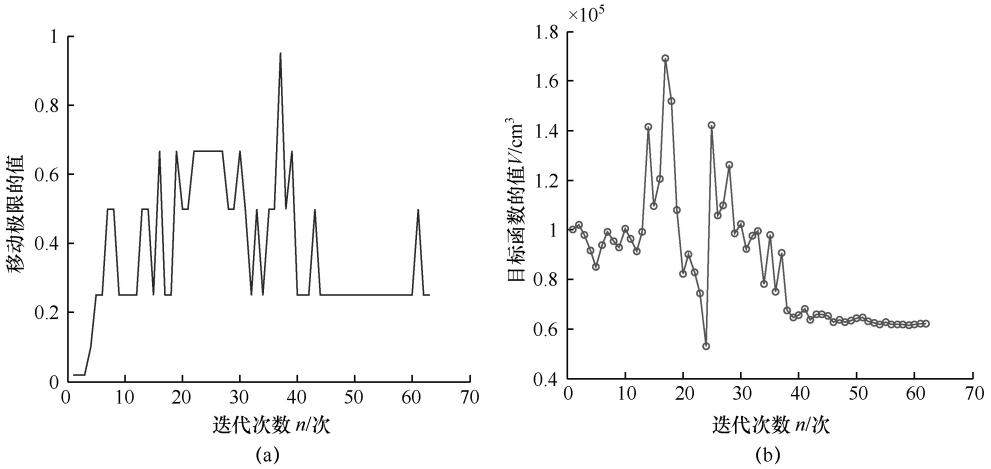


图 5.12　悬臂梁目标函数和移动极限的迭代历史

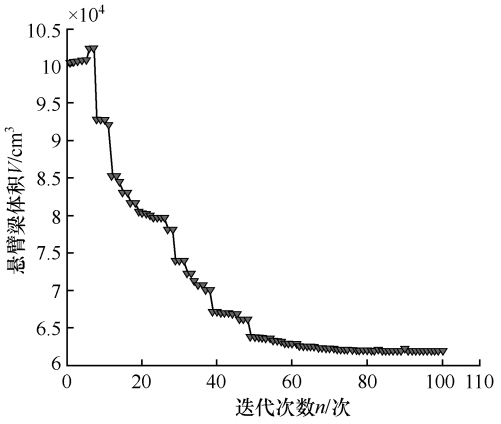


图 5.13　DDSAO 求解阶梯悬臂梁设计问题的优化迭代历程

表 5.2　DDSAO 方法、MLS 方法和文献[56]的比较

	x^*	$f(x^*)$	nit	nfe
DDSAO	（59.832,55.548,50.473,44.083,34.993,2.992,7,2.523,2.205,1.750）	61924.55	101	3328
MLS	（59.83,55.55，50.48,44.11,33.31,2.99,2.78,2.52,2.20,1.87）	62051.748	63	584
文献[56]	（59.84,55.54,50.48,44.08,33.33,2.99,2.78,2.52,2.21,1.92）	62311.940		

5.3.3　变截面悬臂梁形状优化设计

长度为 l=254mm、截面宽度为 b=25.4mm、高度变化的变截面梁自由端受到集中力偶 $M=50.844\text{N}\cdot\text{m}$ 外荷载，发生平面弯曲（如图 5.14 所示）。材料弹性模量 E=68952MPa，泊松比 μ=0.3，保持自由端高度 t=7.62mm 不变。用二维结构固体单元 PLANE42 进行截面变化规律的建模、多变量自适应回归样条与移动极限策略优化。约束条件有：最大应力 $\sigma_{\max}\leqslant$207MPa，最大挠度 $\delta_{\max}\leqslant$12.7mm。模型几何对称，这里取上部分作为计算模型，用 ANSYS 仿真，采用带厚度 PLANE42 平面应力单元，其中 OPT（3）=3。

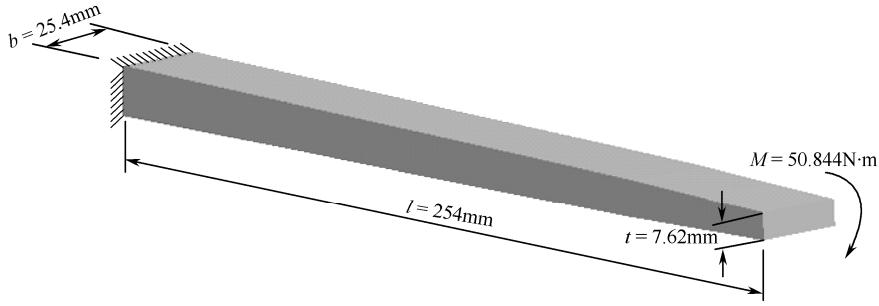


图 5.14　变截面悬臂梁

优化模型为

$$\begin{cases}\min \text{volume}\\ \text{s.t.}\quad \sigma_{\max}\leqslant 207\\ \qquad \delta_{\max}\leqslant 12.7\\ \qquad 0\leqslant \text{dif}_i\leqslant 2.54,\ i=1,2,3\\ \qquad 3.81\leqslant x_j\leqslant 6.86,\ j=1,2,3,4\end{cases}\tag{5.36}$$

式中，dif_i 为保证锥度的约束条件；x_j 为设计变量，即梁沿 x 轴方向的截面高度。

变截面悬臂梁目标函数和移动极限的迭代历史如图 5.15 所示。

初始化构造 MARS 响应面，采用拉丁超立方采样 32 个点，采样点如表 5.3 所示。采样点的 ANSYS 仿真结果如表 5.4 所示。

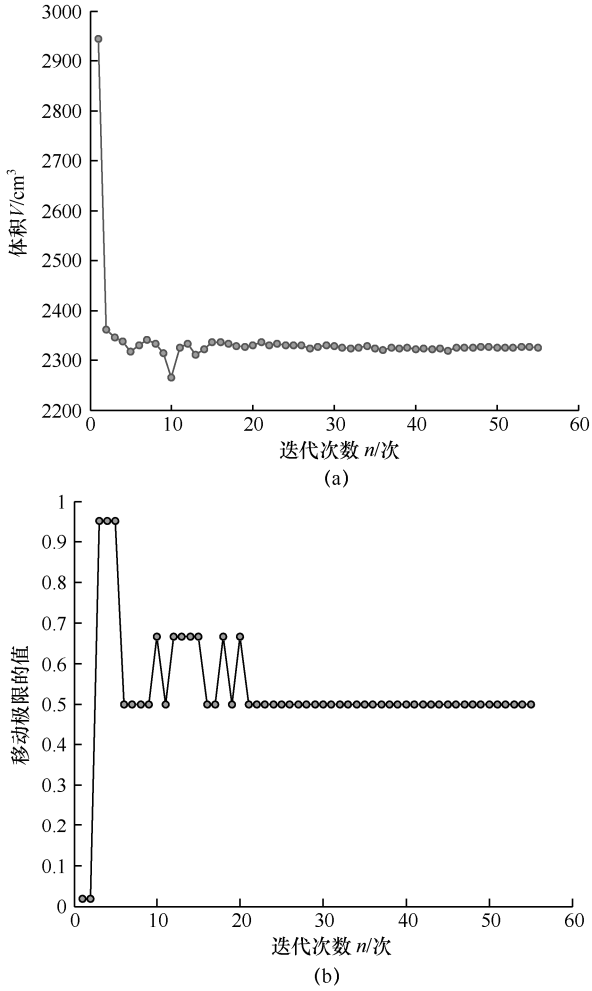


图 5.15　变截面悬臂梁目标函数和移动极限的迭代历史

表 5.3　设计变量的采样点

NO.	x				NO.	x			
	x_1	x_2	x_3	x_4		x_1	x_2	x_3	x_4
1	4.510883	6.221778	4.999245	5.557722	8	6.459986	4.191556	6.064391	4.240036
2	3.980926	4.075411	5.137381	5.284091	9	5.510251	4.600531	5.950777	4.495600
3	4.000764	5.832203	4.655880	4.119133	10	3.896538	5.266841	4.758867	4.118644
4	5.865043	5.167745	6.492611	4.824038	11	4.272573	4.790218	4.038691	4.959039
5	5.750152	6.514015	4.185059	4.752482	12	6.139238	4.379514	5.706438	5.023725
6	4.894998	5.985595	4.282097	6.456747	13	4.703033	4.096695	4.873069	4.289893
7	6.256540	5.499777	5.168366	6.035831	14	4.571032	6.612832	6.197666	5.149842

（续表）

NO.	x				NO.	x			
	x_1	x_2	x_3	x_4		x_1	x_2	x_3	x_4
15	5.137400	5.928064	3.931164	4.067936	24	6.674080	5.043538	5.339141	4.544968
16	4.988663	5.108073	4.125095	6.101530	25	5.305874	6.265684	5.864087	6.610575
17	5.910639	3.931227	5.550100	5.838953	26	6.389210	4.309364	6.675048	4.632498
18	6.582755	4.512327	5.610563	6.553276	27	4.151337	6.328053	6.365274	5.713856
19	6.212926	5.656241	4.549017	5.173944	28	4.742053	5.518901	3.853880	5.463452
20	5.241367	3.859303	6.242623	6.348268	29	4.855351	6.458543	4.916539	3.837902
21	5.662562	4.699448	4.190869	6.273798	30	4.094416	4.154569	6.581349	5.686566
22	5.517265	4.822199	5.266521	5.893484	31	4.411449	6.135957	4.381242	6.180459
23	4.244906	4.954161	5.999924	4.423849	32	5.965355	5.705088	4.447786	3.916263

表 5.4　采样点的 ANSYS 仿真结果

	体积（mm^3）	C1	C2	C3	C4	C5	C6	C7	C8
1	2734.116	–4.11609	1.710895	–4.25089	–1.22253	–1.31747	0.558477	–3.09848	–27.5065
2	2352.689	4.808964	0.094486	–2.63449	1.06197	–3.60197	0.14671	–2.68671	–6.2199
3	2391.59	–0.41282	1.831439	–4.37144	–1.17632	–1.36368	–0.53675	–2.00325	–3.26421
4	2683.448	–3.71998	–0.6973	–1.8427	1.324866	–3.86487	–1.66857	–0.87143	–2.97327
5	2626.308	–3.08026	0.763863	–3.30386	–2.32896	–0.21104	0.567422	–3.10742	–19.3867
6	2776.898	–3.34689	1.090597	–3.6306	–1.7035	–0.8365	2.17465	–4.71465	–43.2099
7	2796.147	–4.57409	–0.75676	–1.78324	–0.33141	–2.20859	0.867465	–3.40747	–32.5046
8	2426.144	0.13101	–2.26843	–0.27157	1.872835	–4.41284	–1.82435	–0.71565	18.62057
9	2471.281	–0.87784	–0.90972	–1.63028	1.350246	–3.89025	–1.45518	–1.08482	0.733281
10	2503.966	–0.50327	1.370303	–3.9103	–0.50797	–2.03203	0.609777	–3.14978	–24.7592
11	2308.449	2.51679	0.517645	–3.05764	–0.75153	–1.78847	0.920348	–3.46035	–21.6018
12	2521.758	–1.11441	–1.75972	–0.78028	1.326924	–3.86692	–0.68271	–1.85729	–11.1116
13	2208.978	4.293317	–0.60634	–1.93366	0.776374	–3.31637	–0.58318	–1.95682	–2.44408
14	2862.804	–4.18002	2.041799	–4.5818	–0.41517	–2.12483	–1.04782	–1.49218	–13.9881
15	2374.136	0.171456	0.790664	–3.33066	–1.9969	–0.5431	0.136773	–2.67677	0.53284
16	2574.128	–0.68883	0.11941	–2.65941	–0.98298	–1.55702	1.976435	–4.51644	–33.3014
17	2550.586	1.011694	–1.97941	–0.56059	1.618873	–4.15887	0.288854	–2.82885	–17.7946
18	2787.402	–2.79898	–2.07043	–0.46957	1.098236	–3.63824	0.942712	–3.48271	–36.3193
19	2620.143	–3.28077	–0.55669	–1.98331	–1.10722	–1.43278	0.624927	–3.16493	–22.9217
20	2660.522	1.967747	–1.38206	–1.15794	2.383319	–4.92332	0.105645	–2.64565	4.702423
21	2711.825	–2.42097	–0.96311	–1.57689	0.741421	–3.28142	0.832929	–3.37293	–33.643
22	2646.793	–2.32701	–0.69507	–1.84493	0.444322	–2.98432	0.626964	–3.16696	–29.1713
23	2460.363	–0.08729	0.709255	–3.24926	1.045762	–3.58576	–1.57607	–0.96393	2.036592

（续表）

	体积（mm^3）	C1	C2	C3	C4	C5	C6	C7	C8
24	2535.928	−2.79843	−1.63054	−0.90946	0.295603	−2.8356	−0.79417	−1.74583	−5.35531
25	3032.45	−6.21193	0.95981	−3.49981	−0.4016	−2.1384	0.746488	−3.28649	−38.0679
26	2566.61	−1.12947	−2.07985	−0.46015	2.365684	−4.90568	−2.04255	−0.49745	7.192856
27	2962.609	−6.22628	0.926716	−3.46672	0.037221	−2.57722	−0.65142	−1.88858	−22.1092
28	2499.857	−0.34946	0.776848	−3.31685	−1.66502	−0.87498	1.609572	−4.14957	−12.3708
29	2513.855	−2.35085	1.603193	−4.14319	−1.542	−0.998	−1.07864	−1.46136	10.61742
30	2778.481	−2.71951	1.310153	−3.85015	1.17678	−3.71678	−0.89478	−1.64522	−19.3039
31	2745.514	−3.2641	1.724508	−4.26451	−1.75471	−0.78529	1.799217	−4.33922	−39.451
32	2412.167	−1.16733	−0.26027	−2.27973	−1.2573	−1.2827	−0.53152	−2.00848	−0.35446

直接用 SQP 优化，仿真采用 ANSYS 的结果如图 5.16 所示。

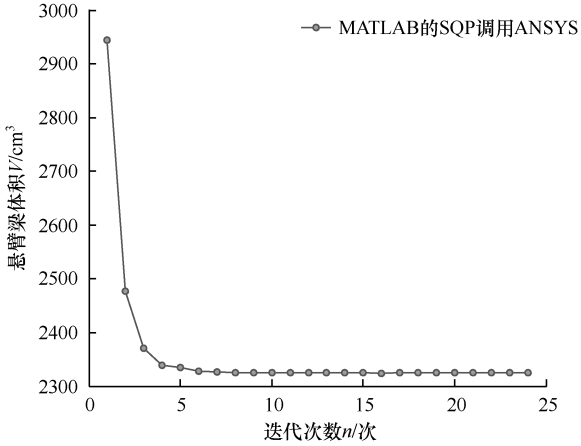


图 5.16　直接用 SQP 优化，仿真采用 ANSYS 的结果

MARS/MLS 方法和 SQP 法计算变截面悬臂梁的外形形状优化结果如表 5.5 所示。

表 5.5　MARS/MLS 方法和 SQP 法计算变截面悬臂梁的外形形状优化结果

	最优点 x^*	体积 V/mm^3	迭代次数 n	仿真次数
SQP	5.523640	2325.814411	24	168
	5.134371	理论值 2323		
	4.578272			
	3.989481			
MARS/MLS	5.197660		55	87
	5.197660	2324.528496		
	4.600322			
	3.986526			

5.3.4 线性两自由度减振器最优设计

这里将第3章中的图3.16所示的线性两自由度减振器最优设计问题应用本章提出的方法求解。初始化构造 MARS 响应面，采用拉丁超立方采样 16 个点，采样点情况如表 5.6 所示。Simulink 模型如图 5.17 所示，模型仿真结果如表 5.7 所示。

表 5.6 线性两自由度减振器问题设计变量的采样点

NO.	f	ξ	NO.	f	ξ
1	1.486363	0.110874	9	1.226786	0.0437
2	1.369911	0.067868	10	0.280441	0.121712
3	1.943402	0.146994	11	0.991158	0.128646
4	0.642328	0.087462	12	0.793748	0.038334
5	0.018662	0.022683	13	1.649574	0.143608
6	1.032189	0.008333	14	0.156385	0.018339
7	1.60509	0.160624	15	0.577006	0.099142
8	0.406785	0.078979	16	1.809161	0.053332

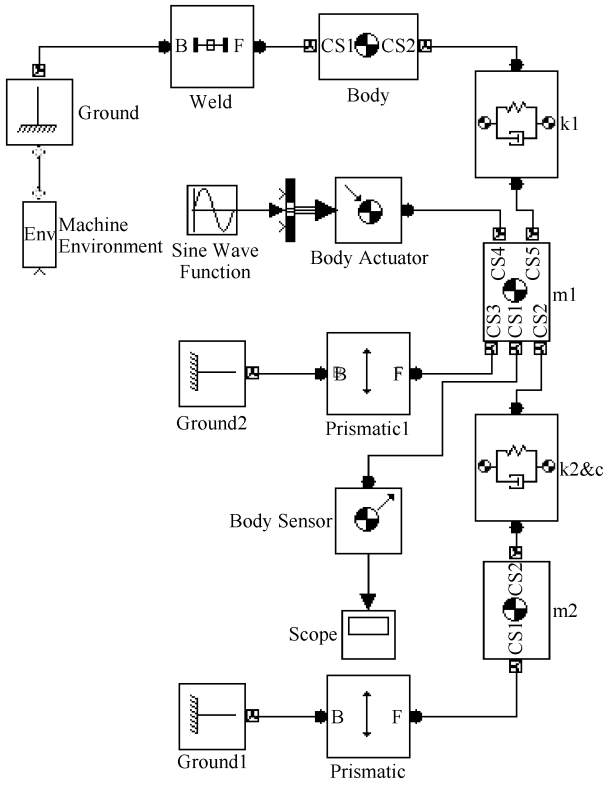


图 5.17 线性两自由度减振器问题的 Simulink 模型

表 5.7　目标函数和约束（与表 5.6 对应）

	d	C1	C2	C3	C4
1	0.073347	–0.42523	–0.30131	–6.42731	0.006359
2	0.064353	–0.65540	–0.70204	–4.80318	–0.21272
3	0.086562	0.029050	0.218951	–8.09919	0.290564
4	0.124932	1.617626	1.729577	–2.29327	1.349315
5	0.125644	1.757619	1.749408	–4.62383	1.071672
6	0.155532	2.756351	2.934312	13.77547	38.71007
7	0.079202	–0.26078	–0.07080	–7.19604	0.133852
8	0.126028	1.699310	1.772719	–3.79984	1.150265
9	0.060459	–0.80871	–0.99404	–1.63842	1.87854
10	0.123434	1.650106	1.670589	–4.24472	1.075327
11	0.082185	–0.0431	0.046615	–1.45541	1.705480
12	0.140970	2.278695	2.360998	1.270077	1.807597
13	0.080442	–0.2323	–0.02201	–7.35024	0.158589
14	0.125841	1.763807	1.765354	–4.60928	1.084789
15	0.124005	1.543336	1.693091	–2.78628	1.258280
16	0.084385	–0.071	0.133235	–7.69042	0.235508

对表 5.6 和表 5.7 中的数据构造 MARS 响应面，得到如图 5.18 和图 5.19 所示的结果。

基于移动极限策略的两自由度问题目标函数迭代如图 5.20 所示。

基于数据驱动的两自由度问题目标函数迭代如图 5.21 所示。

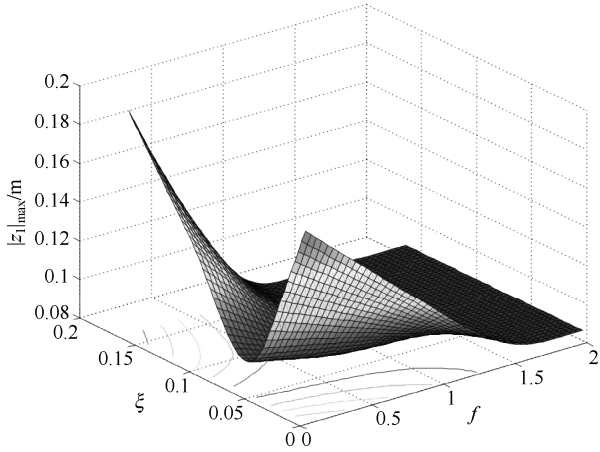
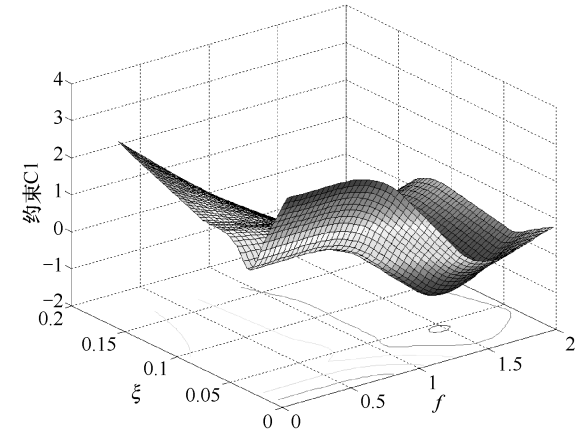
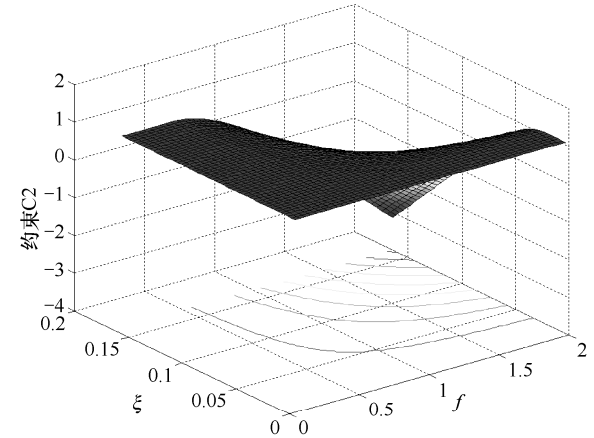


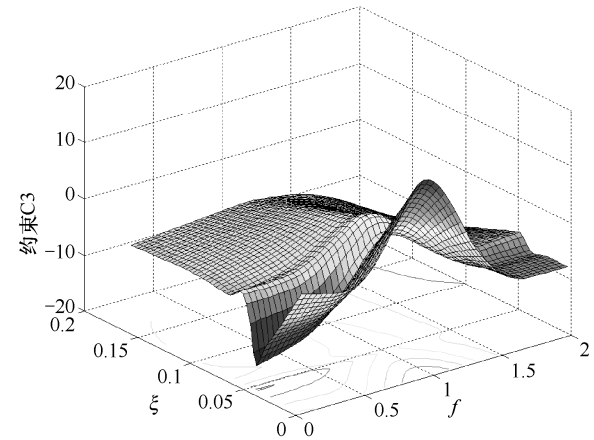
图 5.18　线性两自由度减振器问题目标函数的 MARS 响应面



（a）C1



（b）C2



（c）C3

图 5.19　约束函数的 MARS 响应面

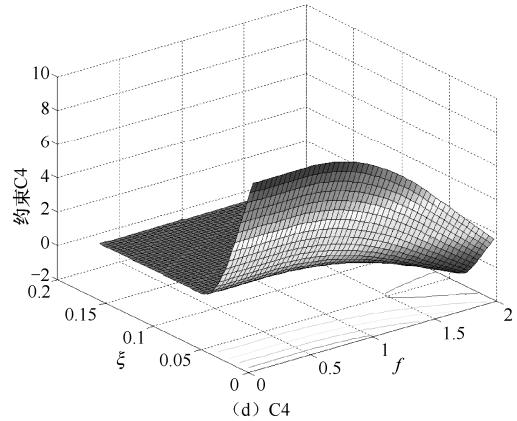


图 5.19　约束函数的 MARS 响应面（续）

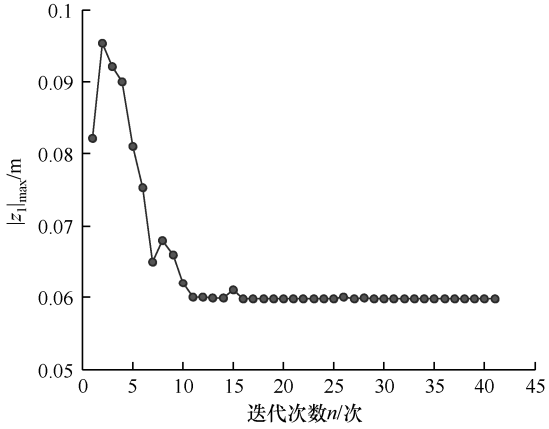


图 5.20　基于移动极限策略的两自由度问题目标函数迭代

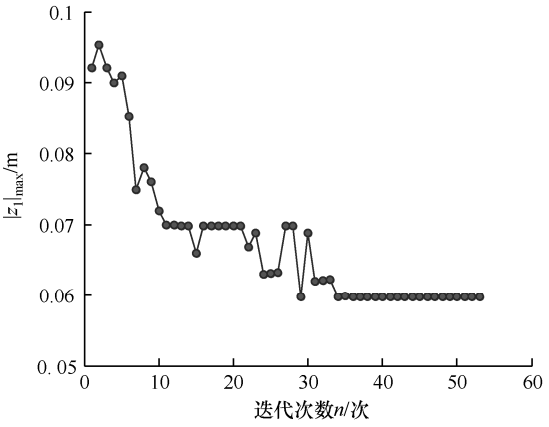


图 5.21　基于数据驱动的两自由度问题目标函数迭代

优化结果比较（两自由度问题）如表 5.8 所示。

表 5.8　优化结果比较（两自由度问题）

		最优点	最大位移响应$\|x_1\|$	迭代次数 n	仿真次数
时间谱元法	固有频率比 f	1.3289	0.059883	34	170
	阻尼系数比 ξ	0.0229			
MLS	固有频率比 f	1.3290	0.05981	41	57
	阻尼系数比 ξ	0.0226			
DDSAO	固有频率比 f	1.3291	0.05993	53	122
	阻尼系数比 ξ	0.0252			

5.3.5　汽车悬架系统动态响应优化设计

这里将第 3 章中的图 3.25 所示的汽车悬架系统动态响应优化设计问题应用本章提出的方法求解。初始化构造 MARS 响应面，采用拉丁超立方采样 56 个点，采样点情况如表 5.9 所示。Simulink 模型如图 5.22 所示，仿真的值（包括约束条件和目标函数）如表 5.10 所示。

表 5.9　线性 5 自由度汽车悬架系统设计变量的采样点

	k_1	k_2	k_3	c_1	c_2	c_3
1	16158.32	125303.9	144269.6	5655.238	1525.047	7249.967
2	70394.49	50692.68	133944.2	4044.018	3260.296	1203.66
3	81133.8	152112	164363.5	2030.222	5768.677	3956.111
4	56477.48	45105.5	114248.5	1812.068	4156.867	7450.1
5	10419.47	107399.4	35470.85	3538.974	11550.75	13057.63
6	64010.78	54364.94	70370.66	1688.829	12575.68	10093.51
7	52586.25	91287.19	62542.7	6244.852	10160.75	3524.927
8	45536.61	58988.75	173052.3	7820.452	10293.68	13149.99
9	80342.75	133189	75135.56	4281.38	1626.138	5206.965
10	54604.5	93714.54	166522.8	3845.892	4723.779	13458.48
11	57377.02	69958.05	119790.8	6068.033	7618.459	4868.791
12	49770.9	131461.6	101746.1	8242.362	10685.56	2011.339
13	71876.29	96372.17	115559.2	5856.718	5282.698	2648.554
14	75773.03	138178.4	73475.15	3434.46	10801.18	11395.88
15	82427.83	156334.7	78708.44	4767.289	5457.223	6117.185
16	47474.66	121647.7	107531.5	747.204	4414.884	11534.71
17	24801.64	171821.4	37918.29	7652.649	4889.074	1591.983
18	59524.66	143592.1	157251.9	6906.822	9585.715	9126.201
19	83678.84	136016.4	59154.47	2847.417	8774.747	6894.681

（续表）

	k_1	k_2	k_3	c_1	c_2	c_3
20	37075.09	100041.9	126031.3	5145.478	2633.577	3298.661
21	11829.63	107671.8	138076.6	7136.475	8425.525	3900.889
22	33017.44	114795.5	46103.8	456.7209	6151.693	2076.061
23	72670.32	167403.6	68754.52	5939.708	3724.776	3218.351
24	84818.35	159605.3	147907.9	6522.602	9332.099	10618.57
25	63502.63	122835.1	81532.42	8547.245	7407.841	6670.483
26	59340.05	85716.34	85627.07	5521.493	2179.824	10022.72
27	22112.44	83398.39	151070.2	4472.026	4377.276	943.1433
28	68549.83	44231.14	146562.9	599.7232	7840.798	2380.116
29	23304.18	140442.4	160745.2	2717.498	6875.914	12008.21
30	49426.26	41833.17	170849	3254.875	13966.78	4338.424
31	20534.03	72819.29	91608.63	2250.921	7179.935	8338.391
32	86312.23	56675.17	83223.07	2364.125	6502.079	5824.555
33	26741.47	111315.5	42091.22	3779.008	2752.103	11706.42
34	27599.18	162064.2	57498.02	1075.711	9053.101	9638.099
35	35834.41	119373.5	136924.1	8344.47	11339.23	7925.242
36	41691.75	79816.72	63405.31	2994.173	1342.36	10848.87
37	44073.6	117311.7	96529.62	5241.108	3579.375	4707.101
38	73676.1	70886.4	42800.22	4634.445	11073.48	7718.601
39	77630.61	49282.32	122353.8	1981.917	3174.475	9368.212
40	43876.13	173119.1	109778.1	5343.561	5854.488	8827.872
41	13787.56	37603.41	132139.5	4901.237	9201.486	8539.439
42	31356.63	66908.32	103226.1	3070.091	6715.969	12712.69
43	9086.697	153860.7	111566.1	4244.028	12033.97	11174.95
44	78209.15	81258.26	50092.32	7502.231	12331.96	6998.163
45	19762.77	104837.2	168684.5	872.5473	13714.24	4568.649
46	14407.03	64070.76	153392.1	7050.971	2420.332	5504.648
47	29925.78	61588.47	65450.8	1182.558	7969.813	10389.16
48	34324.84	89713.14	128039	7952.819	13462.75	8893.319
49	29371.45	102087.2	158671.4	7336.862	12622.4	13681.06
50	52003.9	149049	47769.94	6462.807	1960.62	12438.62
51	40627.62	163074.6	124085.6	8684.623	8301.507	6301.594
52	65681.37	145704.6	88741.15	2600.927	1046.789	12183.59
53	61594.8	77271.87	141841.9	6687.613	11873.56	1554.62
54	17616.88	35096.94	94317.57	1266.781	13067.75	5583.264
55	39356.79	168846.2	99167.67	8022.226	9968.663	13832.41
56	66719.97	128011.9	52621.87	1410.554	13208.28	2764.588

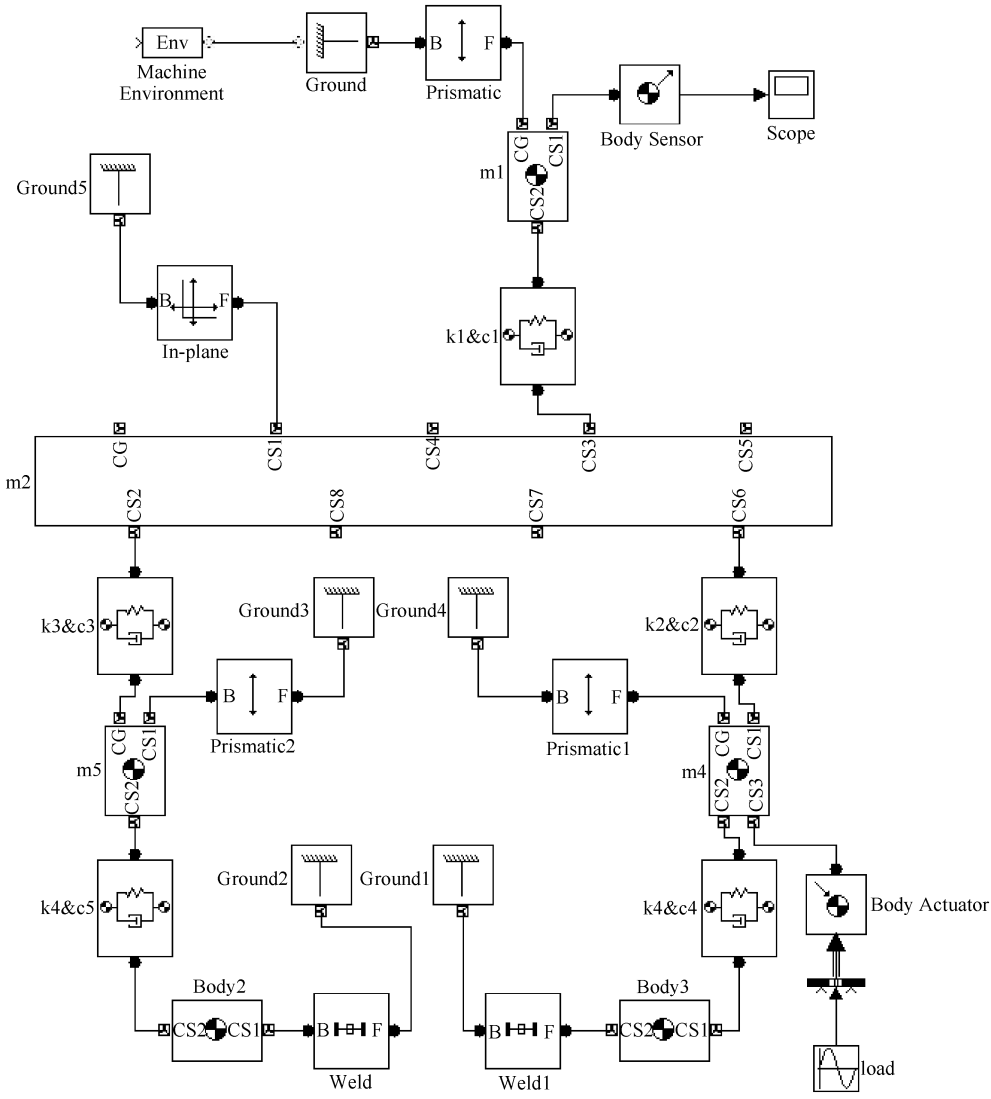


图 5.22　汽车悬架系统设计问题的 Simulilnk 模型

表 5.10　表 5.9 对应的约束条件和目标函数

	C1	C2	C3	C4	C5	*d*
1	–0.01892	–0.02167	–0.08064	0.000116	–0.02147	3.575129
2	–0.03533	0.08259	–0.0851	–0.00367	–0.02807	0.734291
3	0.0364	–0.05275	–0.09101	–0.00565	0.02605	1.999633
4	–0.03282	0.056896	–0.09151	–0.01159	–0.0298	10.91432
5	–0.01128	–0.05476	–0.07573	–0.01124	–0.03175	8.426282

（续表）

	C1	C2	C3	C4	C5	*d*
6	–0.03472	–0.03173	–0.08329	–0.01301	–0.03219	3.339065
7	–0.03722	0.04356	0.05196	–0.01267	–0.02896	5.531793
8	–0.03764	–0.02172	–0.10465	–0.01266	–0.0317	4.700022
9	0.03739	–0.03369	–0.0446	0.00294	–0.02222	2.910938
10	–0.0332	–0.01352	–0.09603	–0.0085	–0.0248	5.136701
11	–0.03786	0.01624	–0.08586	–0.0132	–0.02849	3.135211
12	–0.03818	–0.06153	–0.07612	–0.01263	–0.02881	5.284188
13	0.03764	–0.01539	0.07382	–0.0077	–0.02481	5.850048
14	–0.03803	–0.064	–0.08325	–0.0105	–0.03053	12.46701
15	–0.03941	–0.0604	–0.06085	–0.00969	–0.02628	9.727564
16	0.01968	–0.03702	–0.0821	–0.00792	–0.02444	5.929553
17	–0.03094	–0.06359	0.033732	–0.00828	–0.02456	10.29328
18	–0.03821	–0.06442	–0.0962	–0.0109	0.02907	8.030587
19	–0.03886	–0.06103	–0.0603	–0.01132	–0.02889	6.291412
20	–0.02454	0.002235	–0.07232	–0.00064	–0.02178	8.308691
21	–0.02839	–0.04611	–0.08859	0.01226	–0.02791	6.994023
22	–0.01045	–0.03461	0.001755	–0.0078	–0.02461	7.124908
23	–0.03812	–0.05971	–0.0382	–0.00714	–0.02462	10.48331
24	–0.04054	–0.06803	–0.0951	–0.01065	–0.02899	0.49768
25	–0.03969	–0.05303	–0.06849	–0.01163	–0.02744	9.479731
26	–0.03267	0.020823	–0.07016	–0.00144	–0.02209	2.10894
27	–0.0174	0.007961	–0.08487	–0.00565	–0.02481	6.67691
28	–0.0353	0.006388	–0.09538	–0.01582	–0.03116	6.487585
29	–0.01398	–0.05658	–0.096	–0.01034	–0.02716	12.52549
30	0.03412	–0.03068	0.09939	–0.01366	–0.0311	0.397024
31	–0.01258	–0.01826	–0.08256	–0.01292	–0.02859	10.77266
32	–0.03982	0.003236	–0.07662	–0.01406	–0.02939	11.82904
33	0.01949	–0.02868	–0.06618	–0.0085	–0.02478	11.51898
34	0.000775	–0.06993	–0.0724	–0.01084	0.02965	3.669485
35	–0.03575	–0.05791	0.09405	–0.01093	–0.0303	1.223617
36	–0.02149	0.040887	–0.05904	0.001106	–0.02159	1.437118
37	–0.02988	–0.02735	–0.06247	–0.00536	–0.02338	4.445864
38	–0.03932	–0.03585	0.06132	–0.01295	–0.03221	7.741402
39	0.03603	0.077796	–0.09027	0.00567	–0.02803	9.170422
40	–0.03249	–0.06413	–0.08097	–0.00737	–0.02689	1.633455

（续表）

	C1	C2	C3	C4	C5	*d*
41	–0.02572	–0.00168	–0.09845	–0.01572	–0.03302	12.04642
42	–0.02549	–0.00983	–0.09343	–0.01303	–0.02881	9.935223
43	0.01414	0.07007	–0.09226	–0.01008	–0.03078	4.844367
44	–0.04112	–0.04486	0.06218	–0.01213	–0.03182	4.009645
45	0.016871	0.05659	–0.09765	–0.0113	0.03011	1.149008
46	–0.02282	0.071449	–0.08659	0.000263	–0.02349	11.21356
47	–0.01794	–0.01356	–0.08169	–0.01376	–0.0304	2.672298
48	–0.03542	–0.04986	–0.09448	–0.01099	–0.03089	7.665943
49	–0.03324	–0.05274	0.10217	–0.01037	–0.03087	8.950274
50	–0.03461	–0.04689	–0.07107	–0.00446	–0.02459	4.247739
51	–0.03672	–0.06658	–0.08517	–0.01023	–0.0282	0.050241
52	0.03147	–0.03734	–0.07387	0.00033	–0.02218	2.500061
53	–0.0391	–0.03939	–0.09027	–0.01303	–0.02987	7.310457
54	–0.00478	–0.0267	–0.08483	–0.015	–0.03237	8.686266
55	–0.03571	–0.0719	–0.0916	–0.01002	–0.03008	11.74891
56	–0.03424	–0.06564	–0.0349	–0.01199	–0.02894	10.09437

基于移动极限策略的 5 自由度问题目标函数迭代如图 5.23 所示。

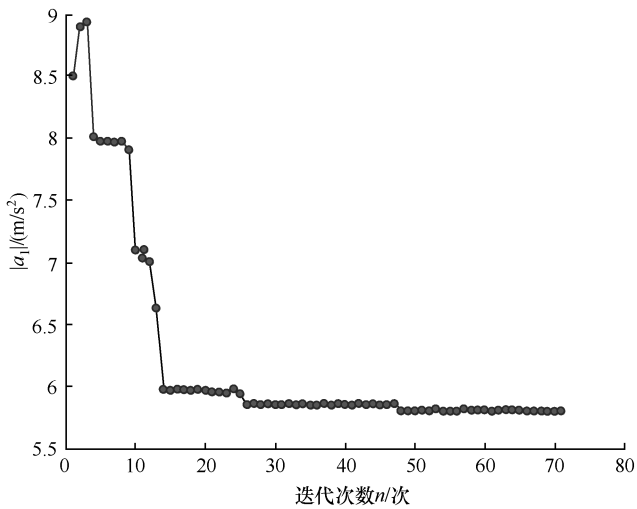


图 5.23　基于移动极限策略的 5 自由度问题目标函数迭代

基于数据驱动的 5 自由度问题目标函数迭代如图 5.24 所示。

优化结果比较（5 自由度问题）如表 5.11 所示。

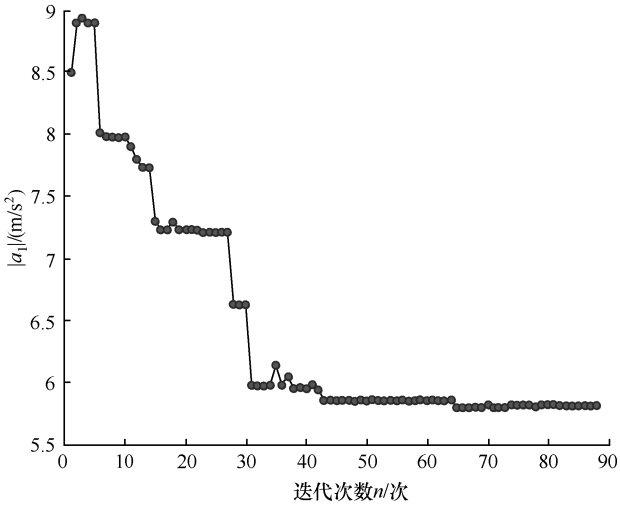


图 5.24　基于数据驱动的 5 自由度问题目标函数迭代

表 5.11　优化结果比较（5 自由度问题）

		最优点	d（目标函数）	迭代次数 n	仿真次数
时间谱元法	k_1	8756.30	5.811567	91	1274
	k_2	35025.20			
	k_3	52879.66			
	c_1	8145.48			
	c_2	8717.84			
	c_3	875.63			
MLS	k_1	8756.30	5.8011	71	127
	k_2	35025.20			
	k_3	52798.64			
	c_1	6342.63			
	c_2	8702.97			
	c_3	875.63			
DDSAO	k_1	8756.30	5.8158	88	232
	k_2	35025.20			
	k_3	53689.82			
	c_1	8133.43			
	c_2	8738.74			
	c_3	875.63			

综上所述，本章用 SQP 和 MARS/MLS 这两种方法计算了变截面悬臂梁的外形形状优化问题，计算结果与理论值的比较见表 5.5。两者的计算结果都和理论值

很接近，但是基于多变量自适应回归样条的移动策略方法非常节约时间，精度也好。序列二次规划法的仿真优化迭代 24 次即达到收敛要求，仿真次数为 168 次；而基于多变量自适应回归样条的移动策略方法迭代 55 次才达到收敛要求，但仿真次数仅仅为 87 次，从这一点来看，它大大节约了计算时间，非常具有工程价值。

本章还用两种方法求解了阶梯悬臂梁截面设计问题，结果如表 5.2 所示，从表 5.2 中可以看出，DDSAO 方法的仿真次数为 3328，而 MLS 的仿真次数为 584，目标函数都接近文献中的值；用两种方法求解了线性两自由度减振器最优设计问题，从表 5.8 中可以看出，时间谱元法的仿真次数为 170，DDSAO 的仿真次数为 122，而 MLS 的仿真次数为 57，目标函数都非常接近精确解；用两种方法求解了 5 自由度汽车悬架系统动态响应优化设计问题，从表 5.11 中可以看出，时间谱元法的仿真次数为 1274，DDSAO 的仿真次数为 232，而 MLS 的仿真次数为 127 次，目标函数都非常接近精确解。从这 3 个算例来看，MLS 方法最具有优势，仿真次数非常少，最适合求解黑箱仿真模型的动态响应优化问题。

5.4 本章小结

本章提出了将基于多变量自适应回归样条（MARS）的全局响应面（GRS）与移动极限策略（MLS），以及将增广拉格朗日方法和有效的数值处理技术有机结合的基于仿真模型的动态响应优化算法。MARS 是一个自适应的回归过程，适合于解决多维问题，它采用了将高维问题简化为低维高精度模型的被修改的回归分块策略。MLS 是在设计空间确定子区间的位置和大小的方法，它不仅反映了函数近似质量，而且反映了优化过程的收敛历史。在后一种方法中，只有当最优点在数据库中，或有很近的点在数据库中时，才用数据库中的点代替，而不需要再进行计算昂贵的仿真分析，如果不存在这样的点，只能通过计算昂贵的仿真分析获得最优点的目标函数和约束函数的值，这样随着优化的进行，数据库越来越成熟。本章提出的方法减少了传统响应面的不利因素，特别是对于高维非线性问题而言。本章应用 GRS/MARS 和 MLS 相结合的方法测试了一个高维函数和一个高维工程问题，表明该方法的可行性和收敛性，并且和二次响应面（QRS）模型在计算效率和精度方面进行了比较，还测试了一个真正调用仿真器 ANSYS 的变截面悬臂梁外形形状优化的问题，并和序列二次规划的仿真优化进行了比较，显示出该方法的优越性。本章将局部 MARS 响应面和增广拉格朗日结合起来改善了序列近似优化，减少了计算昂贵的仿真分析次数，还测试了一些工程问题及一些标准测试函数，如表 5.2、表 5.5、表 5.8 和表 5.11 所示，证明了这两种方法的有效性，也说明了 MLS 方法的优越性。

第 6 章　基于模糊聚类的全局动态响应优化算法

本书的第 3 章和第 4 章提出了针对基于仿真的黑箱模型的动态响应优化算法，然而只是收敛到局部最优。为了解决这个问题，本章又提出一种新的聚类全局动态响应优化算法。由于设计的复杂性，这种黑箱的设计优化软件工具往往需要计算非凸可行域目标函数和约束函数。当对需要计算的目标和约束函数进行分析和仿真优化设计时，搜索算法面临的挑战，以及以多模式模拟目标函数为基础的优化和整体设计的要求对搜索算法提出了更高的要求。在仿真优化中，如果直接使用现有的优化算法，如基于梯度的序列二次规划算法、启发式优化算法、遗传算法（如 GA）等，仿真的数量非常大[87]，往往是工程设计领域难以忍受或不可能实现的。施密特在 1974 年提出响应面近似的优化方法[111、112]，在应用中显示出了巨大的优越性。

近年来，基于元模型的搜索方法成为研究焦点。元模型，也叫做近似模型或代理模型，它具有仿真工具的质量特性和计算密集的黑箱计算机分析能力，并能作为一个设计参数的显函数。用计算速度快的元模型取代密集计算分析和仿真，拥有计算速度快的特点。但是元模型需要有多少个采样点，在什么地方采样才能更好地逼近原函数呢？对于有限的采样点，要构造全局元模型来近似原函数非常困难，而构造局部元模型来近似原函数比较容易，但是在哪个区域构造局部元模型呢？面临如此多的问题，本章将从模糊聚类的角度研究解决，最终解决全局动态响应优化问题。

6.1　模糊数学方法

在自然科学或社会科学研究中，存在许多定义不很严格或具有模糊性的概念。这里所谓的模糊性，主要是指客观事物的差异在中间过渡中的不分明性，如某一生态条件对某种害虫、某种作物的存活或适应性可以评价为“有利、比较有利、不那么有利、不利”；灾害性霜冻气候对农业产量的影响程度为“较重、严重、很严重”等。为处理、分析这些“模糊”概念的数据，便产生了模糊集合论。

根据集合论的要求，一个对象对应于一个集合，要么属于，要么不属于，两者必居其一，且仅居其一。这样的集合论本身并无法处理具体的模糊概念，为处理这些模糊概念而进行的种种努力催生了模糊数学。模糊数学的理论基础是模糊集，模糊集的理论是由美国自动控制专家查德（L. A. Zadeh）教授于 1965 年首先提出来的，在近 10 多年来发展很快。

模糊集合论的提出虽然较晚，但目前在各个领域的应用十分广泛。实践证明，模糊数学在农业中主要用于病虫测报、种植区划、品种选育等方面，在图像识别、天气预报、地质地震、交通运输、医疗诊断、信息控制、人工智能等诸多领域的应用也已初见成效。从该学科的发展趋势来看，它具有极其强大的生命力和渗透力。在侧重于应用的模糊数学分析中，经常应用到聚类分析、模式识别和综合评判等方法。

6.1.1 模糊集的概念

对于一个普通的集合 A，空间中任意一个元素 x，要么 $x \in A$，要么 $x \notin A$，两者必居其一。这一特征可用一个函数表示为

$$A(x)=\begin{cases}1 & x \in A \\ 0 & x \notin A\end{cases} \tag{6.1}$$

$A(x)$即为集合 A 的特征函数。将特征函数推广到模糊集，将在普通集合中只取 0、1 推广到模糊集中为[0, 1]区间。

【定义 1】设 X 为全域，若 A 为 X 上取值为[0, 1]的一个函数，则称 A 为模糊集。

例如，给 5 个同学的性格稳重程度打分，按百分制给分，再除以 100，这样就确定了从 $X=\{x_1,x_2,x_3,x_4,x_5\}$ 到[0, 1]闭区间的一个映射：

x_1：85 分，即 $A(x_1)=0.85$

x_2：75 分，即 $A(x_2)=0.75$

x_3：98 分，即 $A(x_3)=0.98$

x_4：30 分，即 $A(x_4)=0.30$

x_5：60 分，即 $A(x_5)=0.60$

这样确定出一个模糊子集 A=(0.85, 0.75, 0.98, 0.30, 0.60)。

【定义 2】若 A 为 X 上的任一模糊集，对任意 $0 \leqslant \lambda \leqslant 1$，记 $A_\lambda=\{x \mid x \in X, A(x) \geqslant \lambda\}$，称 A_λ 为 A 的 λ 截集。

A_λ 是普通集合而不是模糊集。由于模糊集的边界是模糊的，所以如果要把模糊概念转化为数学语言，需要选取不同的置信水平 λ（$0 \leqslant \lambda \leqslant 1$）来确定其隶属关系。$\lambda$ 截集是把模糊集转化为普通集的方法。模糊集 A 是一个具有游移边界的

集合，则它随 λ 值的变小而增大，即当 $\lambda_1 < \lambda_2$ 时，有 $A_{\lambda_1} \cap A_{\lambda_2}$ 。

【定义 3】若 A、B 为 X 上的两个模糊集，它们的和集、交集和 A 的余集都是模糊集，则其隶属函数分别定义为

$$\begin{aligned}(A \vee B)(x) &= \max(A(x),\ B(x)) \\ (A \wedge B)(x) &= \min(A(x),\ B(x)) \\ AC(x) &= 1 - A(x)\end{aligned} \tag{6.2}$$

关于模糊集的和、交等运算，可以推广到任意多个模糊集合中去。

【定义 4】若一个矩阵元素的取值在[0, 1]区间内，则称该矩阵为模糊矩阵。与普通矩阵一样，有模糊单位阵，记为 $\boldsymbol{I}$；模糊零矩阵，记为 $\mathbf{0}$；元素皆为 1 的矩阵用 $\boldsymbol{J}$ 表示。

【定义 5】若 $\boldsymbol{A}$ 和 $\boldsymbol{B}$ 是 $n\times m$ 和 $m\times 1$ 的模糊矩阵，则它们的乘积 $\boldsymbol{C}=\boldsymbol{AB}$ 为 $n\times 1$ 阵，其元素为

$$C_{ij} = \bigvee_{k=1}^{m}(a_{ik} \wedge b_{kj})(i = 1,2,\cdots,n; j = 1,2,\cdots,m) \tag{6.3}$$

符号“$\vee$”和“$\wedge$”的定义为 $a \vee b = \max(a,b)$，$a \wedge b = \min(a,b)$。

模糊矩阵的乘法性质包括：

（1）$(\boldsymbol{AB})\boldsymbol{C}=\boldsymbol{A}(\boldsymbol{BC})$；（2）$\boldsymbol{AI}=\boldsymbol{IA}=\boldsymbol{A}$；（3）$\boldsymbol{A0}=\mathbf{0}\boldsymbol{A}=\mathbf{0}$；（4）$\boldsymbol{AJ}=\boldsymbol{JA}$；（5）若 $\boldsymbol{A}$、$\boldsymbol{B}$ 为模糊矩阵且 $a_{ij} \leqslant b_{ij}$（一切 i，j），则 $\boldsymbol{A} \leqslant \boldsymbol{B}$，又若 $\boldsymbol{A} \leqslant \boldsymbol{B}$，则 $\boldsymbol{AC} \leqslant \boldsymbol{BC}$，$\boldsymbol{CA} \leqslant \boldsymbol{CB}$。

6.1.2　模糊分类关系

模糊聚类分析是指在模糊分类关系基础上进行聚类。由集合的概念，可给出如下定义。

【定义 6】n 个样品的全体所组成的集合 X 作为全域，令 $X \times Y = \{(X,Y) | x \in X, y \in Y\}$，则称 $X\times Y$ 为 X 的全域乘积空间。

【定义 7】设 R 为 $X\times Y$ 上的一个集合，并且满足：

（1）反身性：$(x_i, y_i) \in R$，即集合中的每个元素和它自己同属一类；

（2）对称性：若 $(x,y) \in R$，则 $(y,x) \in R$，即若集合中的 (x,y) 元素同属于类 R，则 (y,x) 也同属于 R；

（3）传递性：$(x,y) \in R$，$(y,z) \in R$，则有 $(x,z) \in R$。

上述三条性质称为等价关系，满足这三条性质的集合 R 为一个分类关系。

聚类分析的基本思想是用相似性尺度来衡量事物之间的亲疏程度，并以此来实现分类。模糊聚类分析的实质是根据研究对象本身的属性来构造模糊矩阵，在此基础上根据一定的隶属度来确定其分类关系。

6.1.3 模糊聚类

利用模糊理论进行聚类分析的具体步骤如下。

（1）若定义相似系数矩阵用的是定量观察资料，则在定义相似系数矩阵之前，可先对原始数据进行变换处理，变换的方法和系统聚类分析相同。

（2）计算模糊相似矩阵。设 U 是需要被分类对象的全体，建立 U 上的相似系数 R，$R(i,j)$ 表示 i 与 j 之间的相似程度，当 U 为有限集时，$\boldsymbol{R}$ 是一个矩阵，称为相似系数矩阵。定义相似系数矩阵时，原则上可以采用系统聚类分析中的相似系数确定方法，但也可以采用主观评定或集体打分的办法。对数据集

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & x_{1m} \\ x_{21} & x_{22} & x_{2m} \\ \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & x_{nm} \end{bmatrix}_{n\times m}$$

有 8 种建立相似矩阵的方法：①相关系数法；②最大最小法；③算术平均最小法；④几何平均最小法；⑤绝对指数法；⑥绝对值减数法；⑦夹角余弦法；⑧欧氏距离法。

（3）聚类分析。用上述方法建立起来的相似系数矩阵 $\boldsymbol{R}$，一般只满足反射性和对称性，不满足传递性，因而还不是模糊等价关系。为此，需要将 $\boldsymbol{R}$ 改造成 $\boldsymbol{R}^*$ 后得到聚类图，在适当的阈值上进行截取，由此便可得到所需要的分类。将 $\boldsymbol{R}$ 改造成 $\boldsymbol{R}^*$，可采用求传递闭包的方法。$\boldsymbol{R}$ 自乘的思想是按最短距离法原则，寻求两个向量 x_i 与 x_j 的亲密程度。

假设有 $R^2(r_{ij})$，即 $r_{ij}=\overset{n}{\underset{k=1}{\vee}}(r_{ik}\wedge r_{kj})$，说明 x_i 与 x_j 是通过第三者 K 作为媒介发生关系的，$r_{ik}\wedge r_{kj}$ 表示 x_i 与 x_j 的关系密切程度是以 $\min(r_{ik},r_{kj})$ 为准则的。因为 k 是任意的，故从一切 $r_{ik}\wedge r_{kj}$ 中寻求一个使 x_i 和 x_j 关系最密切的通道。$\boldsymbol{R}_m$ 随 m 增加越多，允许连接 x_i 与 x_j 的链的通道就越多。从 x_i 到 x_j 的一切链中，一定存在一个使最大边长达到极小的链，这个边长就相当于 r_{ij}^{∞}。

在实际处理过程中，$\boldsymbol{R}$ 的收敛速度是比较快的。为进一步加快收敛速度，通常采取如下处理方法：

$$\boldsymbol{R}\to\boldsymbol{R}^2\to\boldsymbol{R}^4\to\boldsymbol{R}^8\to\cdots\to\boldsymbol{R}^{2k}$$

即先将 $\boldsymbol{R}$ 自乘改造为 $\boldsymbol{R}^2$，再自乘得 $\boldsymbol{R}^4$，如此继续下去，直到某一步出现 $\boldsymbol{R}^{2k}=\boldsymbol{R}^k=\boldsymbol{R}^*$。此时 $\boldsymbol{R}^*$满足了传递性，于是模糊相似矩阵（$\boldsymbol{R}$）就被改造成了一个模糊等价关系矩阵（$\boldsymbol{R}^*$）。

聚类技术能应用于定量数据，定性数据或两者的混合。这里只考虑处理定量数据。应用最多的是物理过程数据的处理。每一次实验包括 n 个变量的测量，即 n 维行向量 $x_k=[x_{k1},x_{k2},\cdots,x_{kn}],x_k\in R^n$。$N$ 次实验结果可以表示为

$X=\{x_k \mid k=1,2,\cdots,N\}$，即 $N\times n$ 矩阵：

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{bmatrix}$$

在模式识别中，$\boldsymbol{X}$ 的每一行称为模型或对象，每一列称为特征或属性，$\boldsymbol{X}$ 称为模式矩阵，如表 6.1 所示。

表 6.1　模式矩阵和聚类中心

	模式矩阵		决策值		模式矩阵		决策值	聚类中心	
S	a_1	a_2	d	S	a_1	a_2	d	X_1	X_2
u_1	1.926392	1.516378	−1.08788	u_9	2.758663	1.507616	−0.85590	2.866602	1.543089
u_2	2.747509	1.573107	−0.99992	u_{10}	0.388284	1.632682	−0.85076	0.606739	1.584234
u_3	0.195108	1.563268	−0.99772	u_{11}	1.247774	1.532191	−0.94274	2.111808	1.621897
u_4	2.049756	1.678749	−1.11333	u_{12}	0.585023	1.523522	−0.91579		
u_5	3.021129	1.540464	−0.96408	u_{13}	2.090470	1.591263	−1.61016		
u_6	0.697766	1.601620	−0.96151	u_{14}	3.074661	1.551648	−0.98543		
u_7	2.251996	1.604233	−1.71703	u_{15}	2.187521	1.666814	−1.46857		
u_8	0.850898	1.618515	−0.90923						

$u_1,u_2,\cdots,u_{15}$ 是模型或对象，和矩阵中的 $x_1,x_2,\cdots,x_{15}$ 相对应。a_1，a_2 是和 x_{k_1}、x_{k_2} 对应的属性。表 6.1 中包括 Michalewics 函数的 15 个采样点和其函数值及聚类中心。

Gustafson-Kessel 模糊聚类对表 6.1 中数据聚类得到的聚类中心如图 6.1 所示。

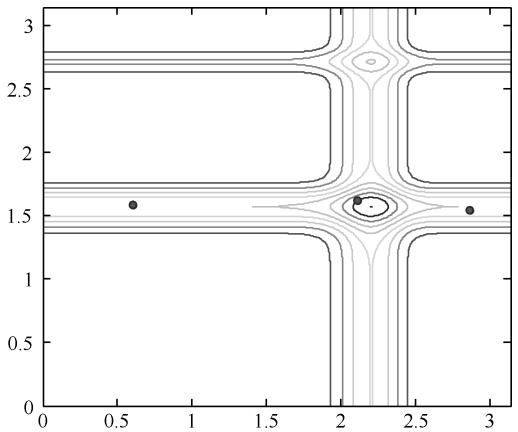


图 6.1　Gustafson-Kessel 模糊聚类对表 6.1 中数据聚类得到的聚类中心

（4）模糊聚类。对满足传递性的模糊分类关系的 $\boldsymbol{R}^*$ 进行聚类处理，给定不同置

信水平的 λ，求 $\boldsymbol{R}_{\lambda}^{*}$ 阵，找出 $\boldsymbol{R}^{*}$ 的 λ 显示，得到普通的分类关系。当 λ=1 时，每个样品自成一类。随着 λ 值的降低，由细到粗逐渐归并，最后得到动态聚类谱系图。

6.2 模糊 C 均值聚类

模糊 C 均值聚类（Fuzzy C-Means，FCM）算法是一种基于划分的聚类算法，它的思想就是使得被划分到同一簇的对象之间的相似度最大，而不同簇之间的相似度最小。模糊 C 均值聚类算法是普通 C 均值算法的改进，普通 C 均值算法对于数据的划分是硬性的，而 FCM 则是一种柔性的模糊划分。

FCM，即众所周知的模糊 ISODATA，是用隶属度确定每个数据点属于某个聚类程度的一种聚类算法。1973 年，Bezdek 提出了该算法，它是对早期硬 C 均值聚类（Hard C-Means，HCM）算法的一种改进。

FCM 把 n 个向量 $x_1(i=1,2,\cdots,n)$ 分为 C 个模糊组，并求每组的聚类中心，使得非相似性指标的价值函数达到最小。FCM 与 HCM 的主要区别在于 FCM 用模糊划分，使得每个给定数据点用值在[0，1]间的隶属度来确定其属于各个组的程度。与引入模糊划分相适应，隶属矩阵 U 允许有取值在[0，1]间的元素。不过，加上归一化规定，一个数据集的隶属度的和总等于 1：

$$\sum_{i=1}^{c} u_{ij} = 1, \forall j = 1, \cdots, n \tag{6.4}$$

因此，FCM 的价值函数（或目标函数）的一般化形式为

$$J(U, c_1, \cdots, c_c) = \sum_{i=1}^{c} J_i = \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{m} d_{ij}^{2} \tag{6.5}$$

式中，u_{ij} 介于[0，1]间；c_i 为模糊组 I 的聚类中心；d_{ij} =‖$c_i - x_j$‖为第 I 个聚类中心与第 j 个数据点间的欧几里得距离；$m \in [1, \infty)$ 是一个加权指数。

构造如下新的目标函数，求使式（6.5）达到最小值的必要条件：

$$\begin{aligned} \overline{J}(U, c_1, \cdots, c_c, \lambda_1, \cdots, \lambda_n) &= J(U, c_1, \cdots, c_c) + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{c} u_{ij} - 1 \right) \\ &= \sum_{i=1}^{c} \sum_{j}^{n} u_{ij}^{n} d_{ij}^{2} + \sum_{j=1}^{n} \lambda_j \left(\sum_{j=1}^{c} u_{ij} - 1 \right) \end{aligned} \tag{6.6}$$

这里的 $\lambda_j(j=1,2,\cdots,n)$ 是式（6.4）的 n 个约束式的拉格朗日乘子。对所有输入参量求导，使式（6.5）达到最小的必要条件为

$$c_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}} \tag{6.7}$$

和

$$u_{ij}=\frac{1}{\sum_{k=1}^{c}\left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \tag{6.8}$$

由上述两个必要条件，可知模糊 C 均值聚类算法是一个简单的迭代过程。当运行批处理方式时，FCM 用下列步骤确定聚类中心 c_i 和隶属矩阵 $\boldsymbol{U}$[87]。

步骤 1：用值在[0，1]间的随机数初始化隶属矩阵 $\boldsymbol{U}$，使其满足式（6.4）中的约束条件。

步骤 2：用式（6.7）计算 C 个聚类中心 c_i，i=1,⋯,C。

步骤 3：根据式（6.5）计算价值函数。如果它小于某个确定的阈值，或它相对上次价值函数值的改变量小于某个阈值，则算法停止。

步骤 4：用式（6.8）计算新的 $\boldsymbol{U}$ 矩阵，然后返回步骤 2。

上述算法也可以采取先初始化聚类中心，然后再执行迭代过程的操作顺序。由于不能确保 FCM 收敛于一个最优解，加上算法的性能依赖于初始聚类中心，所以要么用另外的快速算法确定初始聚类中心，要么每次用不同的初始聚类中心启动该算法，多次运行 FCM。

FCM 算法需要两个参数，一个是聚类数目 C，另一个是参数 m。一般来讲，C 要远远小于聚类样本的总个数，同时要保证 $C>1$。对于 m，它是一个控制算法的柔性参数，如果 m 过大，则聚类效果会很差；如果 m 过小，则算法会接近 HCM 聚类算法。

FCM 算法的输出是 C 个聚类中心点向量和 $C\times N$ 的一个模糊划分矩阵，这个矩阵表示的是每个样本点属于每个类的隶属度。根据这个划分矩阵，按照模糊集合中的最大隶属原则就能够确定每个样本点归为哪个类。聚类中心表示的是每个类的平均特征，可以认为是这个类的代表点。

从 FCM 算法的推导过程中不难看出，它对于满足正态分布的数据聚类而言效果会很好，另外，它对孤立点是敏感的。

6.3　模糊聚类全局仿真优化算法

6.3.1　Kriging 元模型

1. Kriging 方法简介

Kriging 方法是通过已知点来预测位置观察点的一种插值方法。Kriging 方法利用方差的变化来表达空间的变化，而且可以保证由空间分布得到的预测值的误

差最小[114]。

Kriging 方法源于南非的一位矿业工程师 Daniel Gerhard Krige。他率先将统计学应用于地质、矿业的分析与评估。20 世纪 70 年代，法国的数学家 Georges Matheron 对 Krige D.G. 的研究成果进行了系统化、理论化分析，进而提出了一种插值和外推的理论。为了纪念 Krige，将此方法命名为 Kriging 算法。另外，形成了一套新的科学——地质统计学。随着计算机技术的飞速发展，该方法又被运用在计算科学中，用于产生 Kriging 模型，该模型作为一种代理模型可以大大提高计算速度。后来，人们将实验设计方法与使用 Kriging 模型的全过程称为计算机实验设计与分析（DACE）[115]，被广泛应用于多个行业，包括采矿业、水文地质学、自然资源、环境科学、遥感和计算机实验中的黑箱模型[116]。

2. 标准 Kriging 模型

设原函数为 $y(x)$，已知 m 个采样点的自变量和响应值，x 为 n 为空间的自变量 $S_{m\times1}=[s_1,\cdots,s_m]^{\mathrm{T}}$，其中 $s_i\in R^n$。为了方便描述，假设响应值为一维的，即响应值 $Y_{m\times1}=[y_1,\cdots,y_m]^{\mathrm{T}}$，$y_i\in R$。Kriging 模型的回归方程可以表示为[116~118]：

$$\hat{y}(x)=f(x)_{p\times1}^{\mathrm{T}}\beta_{p\times1}+z(x) \tag{6.9}$$

即一个线性回归部分和一个随机过程部分，其中 $f(x)_{p\times1}^{\mathrm{T}}$ 为已知的回归模型的基函数（共 p 个）；β 为对应基函数的系数；$z(x)$ 为一个随机过程，具有以下性质：

（1）$E[z(x)]=0$；

（2）$E[z(\omega)z(x)]=\sigma^2R(\theta,\omega,x)$，其中 σ^2 为该随机过程的方差；$R(\theta,\omega,x)$ 为点 x 和点 ω 之间的关联函数；θ 为关联参数。

1）基函数

回归函数的基函数有多种选法，常用的有[115]：

（1）常数，即 $p=1$，$f_1(x)=1$;

（2）一次函数，$p=n+1$，$f_1(x)=1,f_2(x)=x_1,\cdots,f_{n+1}(x)=x_n$;

（3）二次函数，$p=(n+1)(n+2)/2$，$f_1(x)=1,\ f_2(x)=x_1,\cdots,f_{n+1}(x)=x_n$，$f_{n+2}(x)=x_1^{\ 2},\cdots,f_{2n+1}(x)=x_1x_n,f_{2n+2}(x)=x_2^{\ 2},\cdots,f_{3n}(x)=x_2x_n,\cdots,f_p(x)=x_n^{\ 2}$。

2）关联函数

关联函数模型的基本形式为 $R(\theta,\omega,x)=\prod_{i=1}^{n}R_i(\theta,\omega_i,-x_i)$，常用的关联函数模型有以下几种：

（1）EXP 模型，$R_i(\theta,d_i)=\exp(-\theta_i\mid d_i\mid)$;

（2）EXPG 模型，$R_i(\theta,d_i)=\exp(-\theta_i\mid d_i\mid^{\theta_{n+1}})$，其中 $\theta_{n+1}\in(0,2]$;

（3）高斯模型，$R_i(\theta,d_i)=\exp(-\theta_i\mid d_i\mid^2)$;

（4）线性模型，$R_i(\theta,d_i)=\exp(0,1,-\theta_i\mid d_i\mid^2)$;

（5）球模型，$R_i(\theta,d_i)=1-1.5\varepsilon_i+0.5\varepsilon_i$，其中 $\varepsilon_i=\min\{1,\theta_i\,|\,d_i\,|\}$；

（6）立方模型，$R_i(\theta,d_i)=1-3\varepsilon_i^2+2\varepsilon_i^3$，其中 $\varepsilon_i=\min\{1,\theta_i\,|\,d_i\,|\}$；

（7）样条函数模型，$R_i(\theta,d_i)=\begin{cases}1-15\varepsilon_i^2+30\varepsilon_i^3,\varepsilon_i\in[0,0.2]\\1.25(1-\varepsilon_i)^3,\varepsilon_i\in(0.2,1)\\0,\varepsilon_i\in[1,\infty)\end{cases}$。

关联函数模型及 $\theta=[\theta_1,\theta_2,\cdots,\theta_n]$ 的选取直接影响到 Kriging 模型的精度。模型的选取凭经验，而 θ 的选择可以通过后面的算法实现。

3. 由采样点确定 Kriging 模型

1）系数确定

利用已知的观测点数据，可以得到相关矩阵 $\boldsymbol{R}$（$\boldsymbol{R}$ 为一个对称矩阵）

$$\boldsymbol{R}_{m\times m}=\begin{bmatrix}R(x_1,x_1)&\cdots&R(x_1,x_m)\\\vdots&\ddots&\vdots\\R(x_m,x_1)&\cdots&R(x_m,x_m)\end{bmatrix}$$

及设计矩阵 $\boldsymbol{F}$：

$$\boldsymbol{F}_{m\times m}=\begin{bmatrix}f_1(x_1)&\cdots&f_p(x_1)\\\vdots&\ddots&\vdots\\f_1(x_m)&\cdots&f_p(x_m)\end{bmatrix}$$

根据预测的无偏性可以得到：

$$\hat{\beta}=(\boldsymbol{F}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{F})^{-1}\boldsymbol{F}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{Y}\tag{6.10}$$

式（6.9）表示的 Kriging 模型的方程可以写成下面的形式：

$$\hat{y}(x)=\boldsymbol{f}^{\mathrm{T}}(x)\hat{\beta}+\boldsymbol{r}^{\mathrm{T}}(x)\hat{\boldsymbol{Y}}\tag{6.11}$$

式中，$\boldsymbol{r}_{m\times1}^{\mathrm{T}}(x)=[R(\theta,x,x_1),\cdots,R(\theta,x,x_m)]$；$\hat{\boldsymbol{Y}}=(Y-f\hat{\beta})$。

方差的最大似然估计为

$$\sigma^2=\frac{1}{m}(\boldsymbol{Y}-\boldsymbol{F}\hat{\beta})^{\mathrm{T}}\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{F}\hat{\beta})\tag{6.12}$$

2）θ 向量的最优化选择

使用最大似然估计法来评估 Kriging 模型的参数 $Y=\{\beta,\sigma^2,\theta\}$（回归方程的系数 β，随机过程的方差 σ^2 及空间相关函数的参数 θ），以保证模型响应值与观测值尽可能一致。根据最大似然估计理论，该问题可以转化为一个无约束优化问题[115]：

$$\min\{\psi(\theta)\equiv\sigma(\theta)^2\,|\,R_\theta\,|^{\frac{1}{m}}\}\tag{6.13}$$

根据文献[117]，采用 DERICT 直接搜索算法求解的效率较高。

4. Kriging 方法的特点

优势：作为一种高精度插值模型，Kriging 一般都通过采样点获得数据集；通过选择不同的回归函数和关联函数，Kriging 方法可以灵活地产生不同的实例；此外，Kriging 适用于高阶函数和低维问题，它可以相对少的数据光滑地近似元模型。

不足：由于 Kriging 技术属于插值法，所以它对噪声数据很敏感；对于低阶函数和高维度问题，其效率较低；模型存储量大，如果数据点太多，会导致构造失败；如果相关函数参数的初值设置得不适当，不能产生期望的结果；选择不同的回归函数和关联函数会产生不同的实例，因此一个很实际的问题就是如何选择合适的函数对。

6.3.2 拉丁超立方试验设计

1. LHS

丁超立方采样的通用公式为

$$X_{ij}=[\pi_j(i)-U_{ij}]/n, 1\leqslant i\leqslant n, 1\leqslant j\leqslant d \tag{6.14}$$

式中，n 为采样点的个数；d 为采样点的维度；$\pi_j(1),\cdots,\pi_j(n)$ 是整数 1 到 n 的一个随机排列；$U_{ij}\sim U(0,1)$ 是服从[0, 1]区间的一个均匀随机分布。

拉丁方采样相对于独立同分布来说，有更精确的渐进积分估计：

$$\mathrm{var}_{\mathrm{LHS}}(\tilde{Y})=\frac{1}{n}\int e(x)^2\mathrm{d}F+o(n^{-2}) \tag{6.15}$$

$$\mathrm{var}_{\mathrm{IID}}(\tilde{Y})=\frac{1}{n}\int(f(x)-\mu)^2=\frac{1}{n}(\int e(x)^2\mathrm{d}F+\int\alpha_2^2\mathrm{d}F_2+\cdots\int\alpha_d^a\mathrm{d}F_d) \tag{6.16}$$

对于有界的 f，当 n 趋近于无穷大时，拉丁超立方采样近似分布如下：

$$n^{1/2}(\tilde{Y}-\mu)\to N(0,\int e(X)^2\mathrm{d}F) \tag{6.17}$$

这意味着，对于所有大于等于 1 的整数 p，都有

$$E_{\mathrm{LHS}}(n^{1/2}(\tilde{Y}-\mu))=E_{\mathrm{IID}}(n^{2/2}e)^P+o(n^{-2}) \tag{6.18}$$

2. 栅格采样

把式（6.14）中的均匀分布部分 U_{ij} 用 1/2 替换，得到：

$$X_{ij}=[\pi_j(i)-1/2]/n, 1\leqslant i\leqslant n, 1\leqslant j\leqslant d \tag{6.19}$$

即为栅格采样公式。

6.3.3 算法思想及步骤

初始采样 n（比较小）个点，然后评估黑箱函数得到 n 个函数值点，再构造元模型；在全空间采样 m（比较大，$m\gg n$）个点，然后评估元模型得到 m 个元

模型值，即近似函数值，对其进行从小到大排序，取出 m/100 采样点数的近似函数值所对应的采样点；应用模糊聚类方法对所得采样点 m/100 进行聚类得到聚类中心，然后确定聚类中心的几何中心，再根据聚类中心和几何中心及前两步迭代的结果确定重要区域的半径，最后确定重要区域；在重要区域采样小于等于 n 个点，评估黑箱函数，构造局部 Kriging 模型，再应用 DERICT 优化器进行局部搜索，这样循环直至收敛。对 Michalewics 函数应用以上算法思想时，第一步采样聚类所得的聚类中心如图 6.2 所示，其中 n=32，m=4000。算法流程如图 6.3 所示。下面将详细分析算法的每一个步骤。

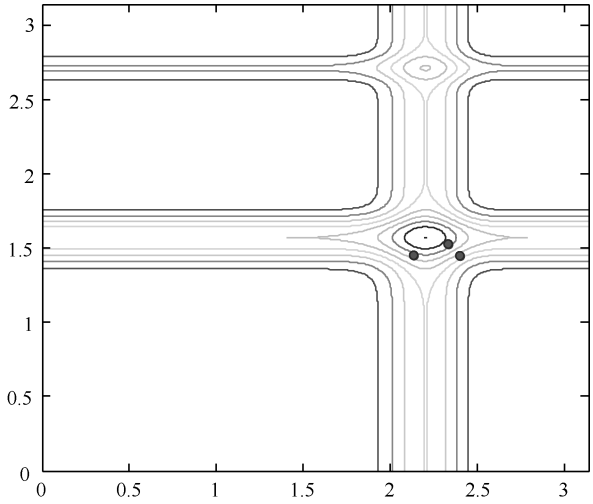


图 6.2　Michalewics 函数的聚类中心

1. 全局采样，取 n 个点，并评估黑箱函数，加入数据库

采用拉丁超立方采样技术，得到 n 个点，然后评估黑箱函数，同时构造一个存储设计变量和黑箱函数值的数据库。因为每一步迭代都要采样，所以随着迭代的进行，数据库越来越成熟，这样构造的元模型就越接近黑箱函数，而且每一次迭代采样后，还要搜索数据库中是否已经存在本次采样点，如果存在，就没有必要再评估黑箱函数，以造成不必要的浪费了。

2. 构造全局 Kriging 模型

在全局区域采样 m ($m \gg n$)个点，仿真 Kriging 模型，将元模型值从小到大排序并取前 m/100 个点。

对于 Kriging 元模型来说，其是数学表达式，仿真一次所需时间极少，因此采样 m 个点，即使采样点数足够大，花费的时间相对仍比较少。

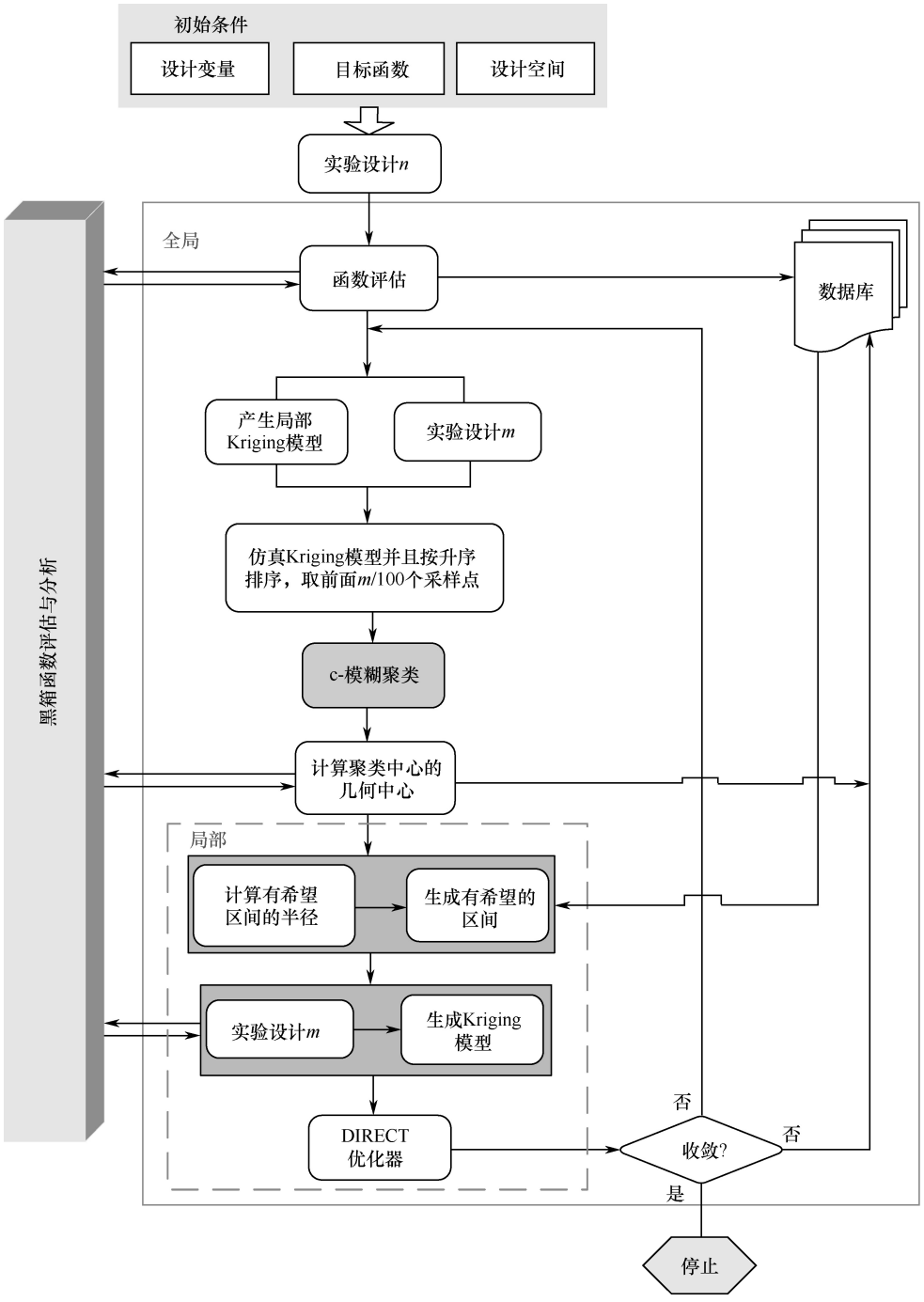


图 6.3　模糊聚类全局仿真优化流程

3. 对 *m*/100 个点进行模糊聚类得到 *C* 个聚类中心，然后评估黑箱函数，将中心点及其黑箱函数值加入数据库中

采用模糊 *C* 均值聚类，聚类中心的个数给定为 3 或 4。首先将数据转化到[0, 1]空间，聚类后再转化到原空间。

4. 取聚类中心的几何中心

几何中心即为

$$\frac{1}{k+1}\sum_{i=1}^{k} x_i \tag{6.20}$$

式中，k 为聚类中心数；x_i 为聚类中心，如果是 n 维空间，则 $x_i=[x_{i_1},x_{i_2},\cdots,x_{i_n}]$。

5. 将聚类中心和聚类中心的几何中心合起来构成有希望的集合 S_p

6. 构造重要区域

对 S_p 中相应的黑箱函数值进行从小到大排序，取出最小值对应的聚类中心点或聚类中心的几何中心，以其为重要区域中心，以 r 为半径构造重要区域。r 通过式（6.21）求得：

$$r^{(k+1)}=\alpha r^{(k)} \tag{6.21}$$

式中，$\boldsymbol{r}^{(0)}=\dfrac{l}{10}$；$l$ 为设计域尺寸向量；α 为重要区域半径系数。

7. 构造局部 Kriging 模型

在重要区域采样 p（$p\leqslant n$）个点，并且搜索数据库找出所有在此区域的点，并和本次采样点合并，删除重复点，评估本次采样点的黑箱函数。

8. 局部搜索

用 DERICT 优化器搜索局部最优点，对所得点进行黑箱函数评估，然后加入数据库中。

9. 收敛准则

如果收敛，则终止迭代；否则转到 2，直至收敛。

在本章提出的方法中，当在可行域的目标函数最小的前 5 个的平均值之差小于 ε 时，即满足式（6.22），优化终止：

$$|\hat{F}_{i+1}-\hat{F}_i|\leqslant\varepsilon,\quad i=1,\cdots,n \tag{6.22}$$

式中，ε 为给定的很小的正数；$\hat{F}_i$ 为

$$\bar{F}_i=\frac{\sum_{j=1}^{5} f_j}{5} \tag{6.23}$$

式中，f_j 为第 j 个最小的函数值。

6.4 实例分析

本章提出的模糊聚类全局优化算法测试了 8 个标准优化问题，并与其他几个全局优化算法进行了比较，包括遗传算法（Genetic Algorithms，GA），模拟退火（Simulated Annealing，SA），粒子群优化（Particle Swarm Optimization，PSO）和模式追踪采样（Mode-Pursuing Sampling，MPS）。这些测试的问题包括以下内容。

1. 六驼峰函数（Six-hump Camel-Back 函数，SC）（n = 2）

$$f_{sc}=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^2,x_1,x_2\in[-2,2] \tag{6.24}$$

2. Griewank 函数（GN）（n = 2）

$$f(x)=\sum_{i=1}^{n}\frac{x_i^2}{200}-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1,x_1,x_2\in[-100,100] \tag{6.25}$$

3. 多项式函数（Generalized polynomial，GF）（n = 2）

$$\begin{aligned}f(x_1,x_2)=&[1.5-x_1(1-x_2)]^2+[2.25-x_1(1-x_2^2)]^2+[2.625-x_1(1-x_2^3)]^2\\&x_1,x_2\in[-2,2]\end{aligned} \tag{6.26}$$

4. Goldstein 价格函数（Goldstein and Price 函数，GP）（n = 2）

$$\begin{aligned}f(x_1,x_2)=&[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times\\&[(30+2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]\\&x_1,x_2\in[-2,2]\end{aligned} \tag{6.27}$$

5. Leon 函数（n = 2）

$$f(x)=100(x_2-x_1^3)^2+(x_1-1)^2,x_1,x_2\in[-10,10] \tag{6.28}$$

6. Himmelblau 函数（Himmelblau，HM）（n = 2）

$$f(x)=(x_1^2+x_2-11)^2+(x_1+x_2^2-7)^2,x_1,x_2\in[-6,6] \tag{6.29}$$

7. Hartmann 函数（n=4）

$$H_{3,4}=-\sum_{i=1}^{4}\alpha_i\exp\left[\sum_{j=1}^{3}A_{ij}(x_j-P_{ij})^2\right],\alpha=[1,1.2,3,3.2]^{\mathrm{T}}$$

$$\boldsymbol{A}=\begin{bmatrix}3.0 & 10 & 30\\ 0.1 & 10 & 35\\ 3.0 & 10 & 30\\ 0.1 & 10 & 35\end{bmatrix},\ \boldsymbol{P}=10^{-4}\begin{bmatrix}6890 & 1170 & 2673\\ 4699 & 4357 & 7470\\ 1091 & 8732 & 5547\\ 381 & 5743 & 8828\end{bmatrix} \tag{6.30}$$

$$0<x_i<1, i=1,2,3$$

$$H_{6,4}=-\sum_{i=1}^{4}\alpha_i\exp\left[\sum_{j=1}^{6}B_{ij}(x_j-Q_{ij})^2\right],\alpha=[1,1.2,3,3.2]^{\mathrm{T}}$$

$$\boldsymbol{B}=\begin{bmatrix}10 & 3 & 17 & 3.05 & 1.7 & 8\\ 0.05 & 10 & 17 & 0.1 & 8 & 14\\ 3 & 3.5 & 1.7 & 10 & 17 & 8\\ 17 & 8 & 0.05 & 10 & 0.1 & 14\end{bmatrix},$$

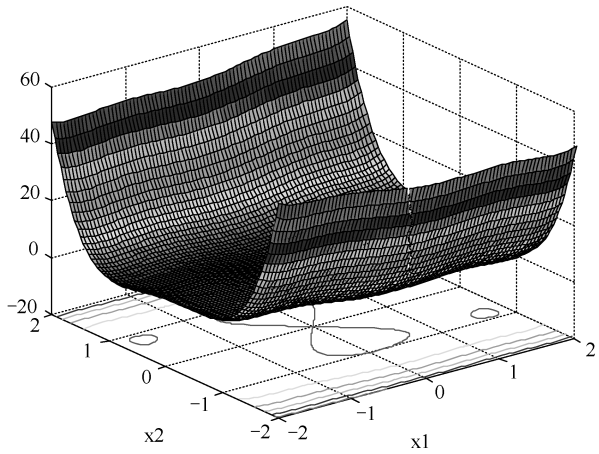
$$\boldsymbol{Q}=10^{-4}\begin{bmatrix}1312 & 1696 & 5569 & 124 & 8283 & 5886\\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991\\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650\\ 4047 & 8828 & 8732 & 5743 & 1091 & 381\end{bmatrix} \tag{6.31}$$

$$0<x_i<1, i=1,2,\cdots,6$$

8. Michalewics 函数（n=2）

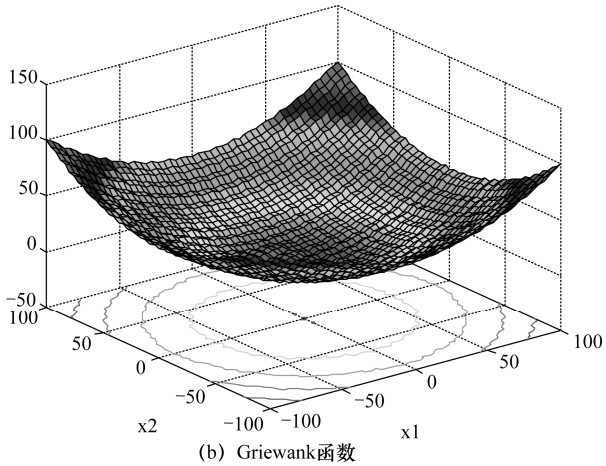
$$f(x_1,x_2)=-\sum^{n}\sin(x_j)[\sin(jx_j^2/\pi)]^{2m}$$
$$m=10, 0\leqslant x_i\leqslant\pi,\ \ i=1,2,\cdots,n \tag{6.32}$$

测试函数的三维图如图 6.4 所示。

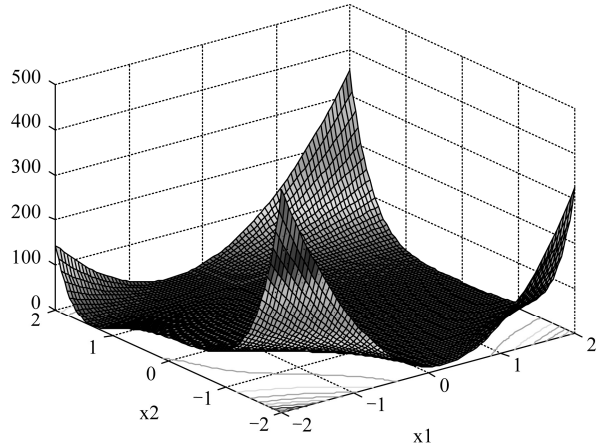


(a) 六驼峰函数

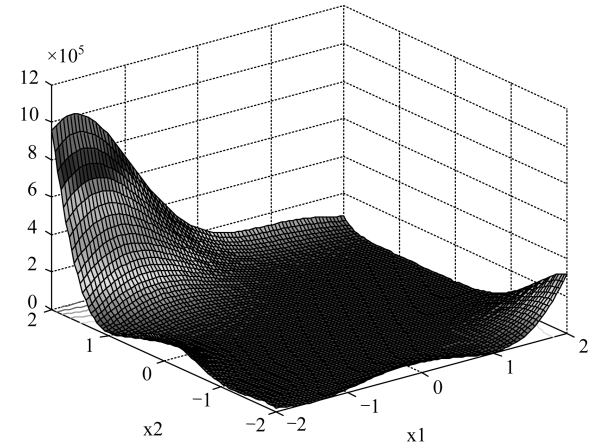
图 6.4　测试函数的三维图



(b) Griewank函数

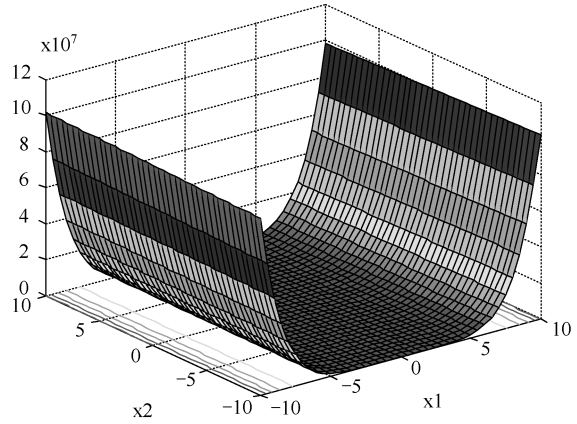


(c) 多项式函数

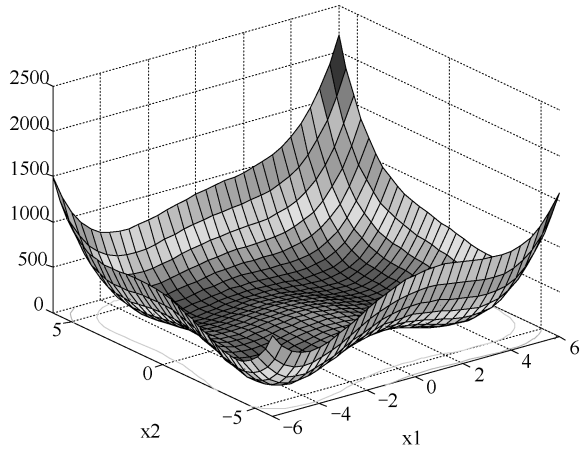


(d) Goldstein价格函数

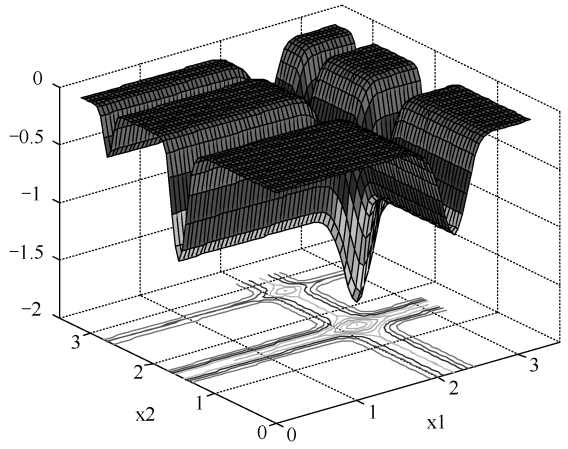
图 6.4　测试函数的三维图（续）



(e) Leon 函数



(f) Himmelblau 函数



(g) Michalewics 函数

图 6.4　测试函数的三维图（续）

9. 线性单自由度减振器最优设计

线性单自由度减振器最优设计的 Simulink 模型如图 6.5 所示。

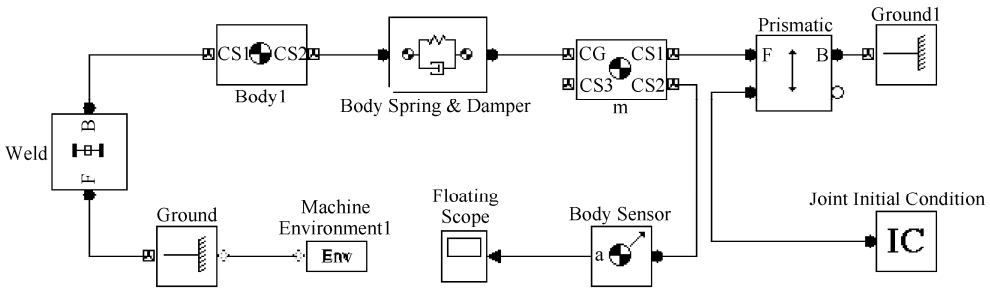


图 6.5　线性单自由度减振器最优设计的 Simulink 模型

为了避免算法的随机性带来的影响，对每一个测试问题分别运行 10 次。在应用精确计算工具的设计优化中，计算时间和黑箱函数的仿真评估与优化迭代次数成正比。因此，在基于仿真的优化方法中，采用函数评估次数（the number of functions evaluations，nfe）和迭代次数（the number of iterations，nit）来评价计算效率。在 FCMO 中，函数评估和迭代次数是对应的，这样如果应用并行计算，迭代次数评价算法的效率是比较合适的[118]。所获得的最小值可以评价算法的精度。然而，如果精度很低，评价算法效率就没有什么意义了。因此，评价一个算法是否有效，前提是能找到精确的最优值点。表 6.2～表 6.12 给出了算法的测试结果。

表 6.2　Six-hump camel-back (SC) 函数 (n=2, $f(x^*)$=−1.0316)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	−1.0316	51	−1.0316	465	−1.0316	2560	−1.0315	13	47	−1.031628	14	93
2	−1.0316	51	−1.0316	470	−1.0316	2452	−1.0312	15	53	−1.031628	14	93
3	−1.0316	51	−1.0316	279	−1.0316	1198	−1.0316	16	58	−1.031628	14	93
4	−1.0316	51	−1.0316	442	−1.0316	1434	−1.0316	11	39	−1.031628	14	93
5	−1.0316	51	−1.0316	434	−1.0316	1240	−1.0313	8	31	−1.031628	14	93
6	−1.0316	51	−1.0316	448	−1.0316	1905	−1.0285	16	46	−1.031628	14	93
7	−1.0316	51	−1.0316	234	−1.0316	1933	−1.0276	9	34	−1.031628	14	93
8	−1.0316	51	−1.0316	426	−1.0316	1217	−1.0314	10	36	−1.031628	14	93
9	−1.0316	51	−1.0316	472	−1.0316	1469	−1.0316	11	39	−1.031608	10	75
10	−1.0316	51	−1.0316	458	−1.0316	1882	−1.0315	8	29	−1.031628	14	93
Mean	−1.0316	51	−1.0316	413	−1.0316	1729	−1.0307	12	41	−1.03163	14	91

表 6.3　Griewank 函数(GN) (n=2, $f(x^*)$=0)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	0.0000	51	0.0000	438	0.0766	1272	0.0001	189	405	0.0002	16	111
2	0.0000	51	0.0000	440	0.1458	1132	0.0000	75	206	0.0002	16	111
3	0.0000	51	0.0000	416	0.1432	2076	0.3613	410	830	0.0002	16	111
4	0.0000	51	0.0001	188	0.1577	1092	9.0181	2	12	0.0002	16	111
5	0.0000	51	0.0000	455	0.1955	1554	0.2076	335	680	0.0002	16	111
6	0.0000	51	0.0000	384	0.1462	1565	1.3253	2	12	0.0002	16	111
7	0.0000	51	0.0000	347	0.0009	1873	0.0185	1	9	0.0002	16	111
8	0.0000	51	0.0000	239	0.1508	2708	0.3871	22	73	0.0002	16	111
9	0.0000	51	0.0000	432	0.0022	1274	0.1970	135	296	0.0002	16	111
10	0.0000	51	0.0000	488	0.1460	2050	0.2061	253	523	0.0002	16	111
Mean	0.0000	51	0.00001	383	0.11649	1660	1.17211	142	305	0.0002	16	111

表 6.4　Generalized polynomial 函数 (GF) (n=2, $f(x^*)$ = 0.5233)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	0.5234	51	0.5233	295	0.5234	1246	1.1061	10	37	0.5233	18	98
2	0.5233	51	0.5233	282	0.5233	1200	0.5285	9	34	0.5233	18	98
3	0.5233	51	0.5233	317	0.5233	1191	0.5233	8	31	0.5233	18	98
4	0.5234	51	0.5233	206	0.5233	1941	0.5234	17	58	0.5233	18	98
5	0.5234	51	0.5233	327	0.5233	1238	0.5351	11	40	0.5233	18	98
6	0.5233	51	0.5233	215	0.5234	1191	0.5237	10	37	0.5233	18	98
7	0.5233	51	0.5233	320	0.5233	1408	0.5233	19	70	0.5233	18	98
8	0.5234	51	0.5233	322	0.5233	1954	0.5247	10	37	0.5233	18	98
9	0.5235	56	0.5233	315	0.5233	1778	1.1239	6	24	0.5233	18	98
10	0.5234	51	0.5233	163	0.5233	2285	1.1003	26	79	0.5233	18	98
Mean	0.52337	52	0.5233	276	0.52332	1543	0.70123	13	45	0.5233	18	98

表 6.5　Goldstein and Price 函数 (GP) (n=2, $f(x^*)$ = 3)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	3.0000	51	3.0000	395	3.0000	1984	3.0001	45	133	3.0000	177	22
2	3.0000	51	3.0000	464	3.0000	1306	29.7997	146	544	3.0000	177	22
3	85.5381	99	3.0147	176	3.0000	1960	3.2758	45	143	3.0000	177	22
4	3.0000	51	3.0073	178	3.0000	1337	90.7282	20	66	3.0000	177	22
5	3.0000	241	3.0000	419	3.0000	1346	3.0000	41	144	3.0000	177	22

（续表）

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
6	3.0003	52	3.0000	447	3.0000	1327	30.0001	32	107	3.0000	177	22
7	3.0002	68	3.0047	159	3.0000	1303	3.0000	42	135	3.0000	177	22
8	3.0000	51	3.0000	460	3.0000	1349	34.7393	134	510	3.0000	177	22
9	3.0000	51	3.0000	443	3.0000	1655	38.2676	9	33	3.0000	177	22
10	3.0000	316	3.0088	216	3.0000	1222	3.2663	24	74	3.0000	177	22
Mean	11.2545	103	3.0035	336	3.0000	1479	23.9077	54	189	3.0000	177	22

表 6.6　Leon 函数(n=2, $f(x^*)$ = 0)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	0.0308	60	0.0079	136	0.0038	1304	4.9316	20	66	0.0000	30	235
2	0.0362	60	0.0000	811	0.0118	3980	2729	27	87	0.0000	30	235
3	0.0276	62	0.0017	375	0.0180	1537	5.7979	16	562	0.0000	30	235
4	0.0112	51	0.0081	270	0.0001	2807	0.4057	27	88	0.0000	30	235
5	0.0509	178	0.0002	556	0.0007	2920	5.5610	41	130	0.0000	30	235
6	0.0589	191	0.0000	872	0.0346	4542	0.6428	20	64	0.0000	30	235
7	0.0861	82	0.0003	280	0.0300	1239	9.1179	11	39	0.0000	30	235
8	0.0470	57	0.0028	285	0.1051	1721	0.2713	31	100	0.0000	30	235
9	0.0287	52	0.0379	175	0.0000	1543	0.0090	52	165	0.0000	30	235
10	0.0019	51	0.0001	306	0.0788	1675	0.6304	33	94	0.0000	30	235
Mean	0.0379	84	0.0059	407	0.0283	2327	3.4341	28	140	0.0000	30	235

表 6.7　Himmelblau 函数 (n=2, $f(x^*)$ = 0)

Runs	GA		PSO		SA		MPS			FCMO		
	min	nit	min	nit	min	nit	min	nit	nfe	min	nit	nfe
1	0.0000	51	0.0000	356	0.0000	1290	0.1642	11	43	0.0000	18	121
2	0.0000	51	0.0000	493	0.0000	1902	0.0605	18	63	0.0000	18	124
3	0.0000	51	0.0000	505	0.0000	1556	0.0091	28	98	0.0000	14	94
4	0.0000	51	0.0000	474	0.0000	1288	0.0039	10	37	0.0000	14	94
5	0.0000	51	0.0000	410	0.0000	2168	0.0001	12	42	0.0000	14	94
6	0.0000	51	0.0000	418	0.0000	1270	0.0415	13	47	0.0000	14	94
7	0.0000	51	0.0004	274	0.0000	1251	0.0074	27	98	0.0000	14	94
8	0.0000	51	0.0000	442	0.0000	1561	0.0656	15	53	0.0000	14	94
9	0.0000	51	0.0054	170	0.0000	2541	0.0335	6	23	0.0000	14	94
10	0.0000	51	0.0000	470	0.0000	1564	0.0022	21	66	0.0000	14	94
Mean	0.0000	51	0.00058	401	0.0000	1639	0.0388	16	57	0.0000	15	100

表 6.8　Hartmann 函数 s（n=2, $f(x^*)$ = −3.86278）

Runs	GA			SA			FCMO		
	min	nit	nfe	min	nit	nfe	min	nit	nfe
1	−3.86278	51	1040	−3.8626	3840	3898	−3.8439	12	98
2	−3.86278	51	1040	−3.8625	4620	4690	−3.8439	12	98
3	−3.86278	51	1040	−3.8606	2570	2607	−3.8439	12	98
4	−3.86278	51	1040	−3.8627	1870	1898	−3.8439	12	98
5	−3.86278	51	1040	−3.8625	3060	3106	−3.8439	12	98
6	−3.86278	51	1040	−3.8624	1900	1928	−3.8439	12	98
7	−3.86278	51	1040	−3.8561	1670	1695	−3.8439	12	98
8	−3.86278	51	1040	−3.8591	1870	1898	−3.8439	12	98
9	−3.86278	51	1040	−3.8625	4070	4131	−3.8439	12	98
10	−3.86278	51	1040	−3.8621	3240	3289	−3.8439	12	98
Mean	−3.86278	51	1040	−3.8613	2871	2914	−3.8439	12	98

表 6.9　Hartman 函数(n=6, $f(x^*)$ = −3.322)

Runs	GA		MPS			FCMO		
	min	nit	min	nit	nfe	min	nit	nfe
1	−3.3195	54	−3.322	104	959	−3.2945	26	218
2	−3.2019	80	−3.3218	120	1107	−3.3013	38	357
3	−3.2014	51	−3.1957	171	1563	−3.2945	26	218
4	−3.2831	52	−3.3185	70	654	−3.2945	26	218
5	−3.3103	75	−3.3216	96	889	−3.2945	26	218
6	−3.3187	79	−3.3065	72	672	−3.2945	26	218
7	−3.3087	60	−3.322	89	826	−3.2945	26	218
8	−3.3083	76	−3.3193	66	618	−3.2945	26	218
9	−3.2004	53	−3.1297	143	1293	−3.2945	26	218
10	−3.3160	60	−3.3212	77	718	−3.2945	26	218
Mean	−3.2768	64	−3.2878	101	930	−3.2952	27	232

表 6.10　Michalewics 函数(n=2, $f(x^*)$ = −1.8013)

Runs	GA			MPS			FCMO		
	min	nit	nfe	min	nit	nfe	min	nit	nfe
1	−1.801	51	1040	−1.79798	1500	1517	−1.8013	12	98
2	−1.801	51	1040	−1.80125	2400	2427	−1.8013	14	113
3	−1.801	51	1040	−1.80129	1580	1597	−1.8013	14	111
4	−1.801	51	1040	−1.80126	1480	1497	−1.8013	14	113
5	−1.801	51	1040	−1.80122	2080	2103	−1.8013	14	113

（续表）

Runs	GA			MPS			FCMO		
	min	nit	nfe	min	nit	nfe	min	nit	nfe
6	−1.801	51	1040	−1.80125	2910	2943	−1.8013	14	113
7	−1.801	51	1040	−1.8013	1660	1679	−1.8013	14	113
8	−1.801	51	1040	−1.80129	1530	1547	−1.8013	14	113
9	−1.801	51	1040	−1.80117	1890	1911	−1.8013	14	113
10	−1.801	51	1040	−1.80127	2700	2731	−1.8013	14	113
Mean	−1.801	51	1040	−1.44067	1973	1995	−1.8013	14	111

表 6.11　Michalewics 函数(n=5, $f(x^*)$ = −4.687658)

Runs	GA			MPS			FCMO		
	min	nit	nfe	min	nit	nfe	min	nit	nfe
1	−4.5376	56	1140	−3.83676	3250	3331	−4.5369	34	264
2	−4.5376	78	1580	−3.9338	4120	4236	−3.5981	50	380
3	−4.4530	77	1560	−3.35024	2500	2566	−4.2762	18	164
4	−4.4959	85	1720	−4.2828	5800	5961	−4.2762	18	164
5	−4.5376	96	1940	−3.05837	3600	3696	−4.6845	46	511
6	−3.9504	78	1580	−4.04446	2700	2766	−4.6845	46	511
7	−4.5376	91	1840	−4.16786	4330	4446	−3.5941	18	224
8	−4.6459	98	1980	−3.71568	3240	3326	−4.6845	46	311
9	−4.6875	96	1976	−3.67189	3950	4056	−4.6845	46	311
10	−4.4953	99	2000	−2.65524	3740	3846	−4.6845	46	311
Mean	−4.4878	85	1732	−3.67171	3723	3823	−4.3704	37	315

表 6.12　线性单自由度减振器最优设计(n=1, $f(x^*)$ = 0.5206)

Runs	GA			MPS			FCMO		
	min	nit	nfe	min	nit	nfe	min	nit	nfe
1	0.5205984	56	940	0.5209936	325	383	0.5205983	46	311
2	0.5205984	56	940	0.5209956	420	439	0.5205983	46	311
3	0.5205984	56	940	0.5206213	200	282	0.5205983	46	311
4	0.5205984	56	940	0.5206652	810	962	0.5205983	46	311
5	0.5205984	56	940	0.5205992	603	768	0.5205983	46	311
6	0.5205984	56	940	0.5206213	370	421	0.5205983	46	311
7	0.5205984	56	940	0.5206562	567	684	0.5205983	18	311
8	0.5205984	56	940	0.5206219	368	417	0.5205983	46	311
9	0.5205984	56	940	0.5205987	406	561	0.5205983	46	311
10	0.5205984	56	940	0.5206236	380	396	0.5205983	46	311
Mean	0.5205984	56	940	0.5206330	363	393	0.5205983	46	311

因为迭代次数和函数评估次数是正整数，所以 10 次运行的平均值是经过圆整的。从总体上来说，FCMO 的平均值都比较小，而且 FCMO 更精确、更稳定。对于 GN 函数来说，只有 FCMO 找到了更小函数值的点，GA 和 PSO 也找到了较小的函数值的点，而 MPS 最差。对于所有的测试函数来说，FCMO 获得了最满意的结果。

尽管对于有的测试函数来说，FCMO 函数的仿真次数和迭代次数比 MPS 大一些，但从算法精度和稳定性来比较，FCMO 算法最好。对于 GA 和 PSO 来说，虽然它们也有一定的精度和稳定性，但是有非常大的仿真次数和迭代次数，在仿真优化工程中，应用起来很难被人们接受。而 SA 不仅函数仿真的评估次数和迭代次数大，而且精度相对比较差。

从测试的总体来说，FCMO 算法不仅对低维问题（如 2、3 维问题），而且对高维问题（如 5，6 维问题）都获得了满意的结果，这说明 FCMO 算法具有普遍的实用性。

本章中提出了基于模糊聚类和元模型的全局优化算法。其优点是精度高、稳定、迭代次数少。它尤其适合密集型计算、黑箱分析与仿真和高维优化问题。在本章提出的方法中，元模型的选择是至关重要的，这不仅和优化设计问题有关，而且和元模型的属性有关。当然，可以考虑应用混合元模型。对于同一问题，构造不同的元模型所需要的时间不同，如某一个 16 维的问题，采样 374 个点，构造一个 Kriging 模型需要 64.61s（Pentium D CPU，3.0GHz，1G RAM），而构造一个 MARS 元模型需要 90.52s。尽管构造元模型花费的时间和评估黑箱函数花费的时间相比很少，但是随着迭代次数的增加，数据点越来越多，这样构造元模型的时间也相当可观。因此本章采用了一个元模型而没有采用混合元模型。

FCMO 方法选择了 Kriging 元模型，并根据前两次重要域和半径决定当前步的重要域和半径。应用 FCMO 是为了寻找最有希望包括全局最优点的区域或其附近区域，然后调用 DIRECT 优化器快速搜索重要域的最小值点。通过经典的全局优化测试函数，包括从低维 2D、3D 到高维 5D、6D，结果显示 FCMO 算法的计算效率高，稳定性好。

由于 FCMO 方法包括额外的计算，如需要模糊聚类运算，所以本章提出的算法处理一些简单的低维问题时的效率比其他全局优化算法的效率低。在不同条件下，采用不同的采样机制，将进一步提高 FCMO 方法的效率。FCMO 方法为进一步研究工程应用奠定了基础，如基于 FEM 或 CFD 的优化，下一步可将 FCMO 方法扩展到约束优化问题。

6.5 本章小结

对于昂贵计算和黑箱函数优化问题，研究的重点在于减少黑箱函数的评估和计算昂贵分析的次数。本章应用了 Kriging 模型，这是因为在优化过程中，需要反复进行目标函数的评估，而近似模型是数学模型，评估一次近似模型需要的时间很少。当然，黑箱函数必须要评估，但是评估次数一定要非常少。当近似模型非常接近黑箱函数时，我们就可以获得近似模型的局部峰值，然后在这个局部区域采样，评估黑箱函数，构造局部近似模型，最后进行局部搜索。反复进行这个过程直到找到全局最小值为止。本章还提出了模糊聚类全局优化：每一次迭代用聚类方法找到 3 个聚类中心（当然也可以更多），这 3 个中心就是局部最优或其附近，再找到这 3 个聚类中心的几何中心；然后评估黑箱函数，获得这 3 个点的函数值；最后以最小值点为重要区域中心，在重要区域中采样，建立近似模型并优化求解。对于本章提出的方法，测试了 10 个标准全局优化问题和一个带约束的动态响应优化问题，并且和遗传算法（GA）、模拟退火算法（SA）、粒子群算法（PSO）及模式追踪算法（MPS）进行了比较，证明了本章提出方法的精确性和鲁棒性。

附录 A　动态响应优化问题的并行化处理程序

```
/*********************************************************/
/* 函数名称： cvm01_func                                  */
/* 功能： 计算目标函数和约束函数*/
/*********************************************************/
void cvm01_func(int n,int m,int nc,int *jg,double *x,double
                *g,double *varlb,
                            double *varub)
{
    //一个任务也可以并行，找到网络上最快的计算机执行
    double  f,*d1;
    int n0=0,n1=0;
    int i;
    double  *x0;

    x0 = new double[n];
    for (i=0; i<n; i++)
    {
        x0[i] = x[i+1];
    }
    //d1 = alloc(nc);
    d1 = new double[nc];
    //if (sfd=='d') transfnt(x0,&f,d1,n,nc);

    //由设计变量的一组值，评估目标和约束值
BOOL  bParallel  =  m_pOptData  ->  GetParallelCompute()  &&
                    ParallelCompute::m_bParalled;
    double ** X;
    double *  F;
    double ** G;
```

```
if (bParallel)//并行
{
    X = new double*[1];
    F = new double[1];
    G = new double*[1];
    X[0] = x0;
    F[0] = 0; //f
    G[0] = d1;

    vector<SimulateData> simDatas = CreateSimData(/*m_
    pOptData, */1, X /*, F, G*/);
    ParallelCompute::ParallelSimulating(simDatas);
    SetSimData(simDatas, F, G);
    simDatas.clear();

    f = F[0];
    //for (int i=0; i<nc; i++)
    //{
    //  d1[i] = G[0][i];
    //}
}
else {
    Evaluate(x0, &f, d1);
}

//fprintf(fp, "in func call:\n");
//for (i=0; i<n; i++)
//{
//  fprintf(fp, "x0[%d]=%.8f; ", i, x0[i]);
//}

//fprintf(fp, "\nf=%.8f\n", f);
//for (i=0; i<nc; i++)
//{
//  fprintf(fp, "d1[%d]=%.8f; ", i, d1[i]);
//}
//fprintf(fp, "\n");

//wyz add 2007,6,5
```

```
for (i=0; i<nc; i++)
{
    if (d1[i] > 0)
    {
        f += d1[i] * 1E+16;
    }
}

for (i=1; i<=m; i++)
{
    switch (jg[i])
    {
    case 1:
        g[i] = f;
        n0 = 1;
        break;

    case 2:
        g[i] = d1[n0-1];
        n0 = n0+1;
        break;

    case 3:
        g[i] = -d1[n0-1];
        n0 = n0+1;
        break;

    default:
        if (jg[i] > 0)
        {
            n1 = jg[i]-3;
            g[i] = x[n1]-varlb[n1];
        }
        else
        {
            n1 = abs(jg[i])-3;
            g[i] = varub[n1]-x[n1];
        }
        break;
```

```
            }
        }

        OPB_nf = OPB_nf+1;
        delete [] x0;
        delete [] d1;

        if (bParallel)
        {
            delete [] X;
            delete [] F;
            delete [] G;
        }
    }
```

附录 B　基于时间谱元法的动态响应优化算法程序

1. 基于 GLL 点法的动态响应程序

```
function [history,searchdir] = fiveDegMSD_OPT_ProblemOne(arg1,
                                arg2)
%%
global GLL_allData_No7colom;
history.x = [];
history.fval = [];
searchdir = [];

global ne inp;
global s ig;%gdm gdm2 bf
ig = 1;

s=2.0;%时间上限
% ne=10;inp=3;
ne = arg1;
inp = arg2;

% x00 前面三项分别是 k,c,b3（人工变量）
% k1=17512.6;k2=52537.8;k3=52537.8;
% c1=1751.26;c2=4378.15;c3=4378.15;
x00 = [17512.6 52537.8 52537.8 1751.26 4378.15 4378.15 8.448];
      % Starting guess at the solution
%x00 = [150; 0.02; 0.01];

lb=[8756.30 35025.20 35025.20 350.252 875.630 875.630 0.0254];
ub=[87563.0 175125.98 175125.98 8756.3 14010.08 14010.08 12.7];


opts=optimset('Algorithm','active-set','outputfcn',@outfun,'
```

```
                MaxFunEvals',5000,'MaxIter',500);
opts=optimset(opts,'TolFun',1.0e-8,'TolX',1.0e-15,'TolCon',1.
                0e-15,'Display','iter');
%opts=optimset(opts,'GradObj','on','GradConstr','on');
A=[];b=[];Aeq=[];beq=[];
t=cputime;
[x,fv,exitflag,ot] = fmincon(@fiveDegObj,x00,A,b,Aeq,beq,lb,
                      ub,@fiveDegConFun,opts)
tcpu=cputime-t

% 优化结果写文件
fp=fopen(':\ \OptResultCriticalPoint_fivedeg_1.txt','a');
fprintf(fp,'****************************************\r\n');
fprintf(fp,'单元数 Nel=%d，插值点数 interp=%d\r\n',ne,inp);
fprintf(fp,'****************************************\r\n');
fprintf(fp,'%0.6f\r\n',x);
fprintf(fp,'iteration = %d\r\n',ot.iterations);
fprintf(fp,'t = %0.6f\r\n\r\n',tcpu);
fclose(fp);

if ot.funcCount<opts.MaxFunEvals && ot.iterations<opts.MaxIter
   disp（'正常结束!'）;
   iter = ot.iterations;
   x3x=history.x;
   f=history.fval;
   xx=0:iter-1;
else
   disp（'函数评估次数超过最大值!'）;
   iter = ot.iterations;
   x3x=history.x;
   f=history.fval;
   xx=0:iter;
end
%

%    subplot(2,2,1);

plot(xx,x3x(:,7),'-or','MarkerFaceColor','g');
xlabel('迭代次数  n/次');ylabel('目标函数|y1|/m/s^2');box
```

```
off;set(gcf,'Color','w');
legendname = sprintf('GLL 点法: Nel=%d,m=%d',ne,inp); h =
             legend(legendname);set(h,'Interpreter','none');
name = sprintf('E: \\CriticalPoint_OneDeg\\FiveDeg_GLL_obj_
               Nel%dm%d.fig',ne,inp);
saveas(gcf,name);close all;

GLL_allData_No7colom = x3x;
matname = sprintf('E:\\CriticalPoint_OneDeg\\matfile\\fiveDeg_
                GLL_Nel%dm%d.mat',ne,inp);
save(matname,'GLL_allData_No7colom');

    function stop = outfun(x,optimValues,state)
        stop = false;

        switch state
            case 'init'
                %hold on
            case 'iter'
                % Concatenate current point and objective function
                % value with history. x must be a row vector.
                history.fval = [history.fval; optimValues.fval];
                history.x = [history.x; x];
                % Concatenate current search direction with
                % searchdir.
                searchdir = [searchdir;...
                    optimValues.searchdirection'];
                %plot(x(1),x(2),'o','MarkerFaceColor','g');
                %Label points with iteration number and add title.
                %text(x(1)+.015,x(2),...
                    %num2str(optimValues.iteration));
                %title('Sequence of Points Computed by fmincon');
            case 'done'
                %hold off
            otherwise
        end
    end
end
```

```
% 目标函数
function f=fiveDegObj(x)
f=x(7);
end

% 约束函数
function [C,Ceq]=fiveDegConfun(x)
global s ne inp;
% global gdm gdm2;

k1=x(1);
k2=x(2);
k3=x(3);
c1=x(4);
c2=x(5);
c3=x(6);
d=x(7);

[xg,uva]=fiveDegUVA(s,ne,inp,k1,k2,k3,c1,c2,c3);
ng=length(xg);

dd=d*ones(1,ng);

%加速度小于人工变量
C1=uva(16,:)'-dd';
C2=-uva(16,:)'-dd';

%相对位移约束
L=3.048381;%车长
%车体与司机
C3=uva(2,:)'+L/12*uva(3,:)'-uva(1,:)'-ones(ng,1)*0.0508;
C4=-(uva(2,:)'+L/12*uva(3,:)'-uva(1,:)')-ones(ng,1)*0.0508;
%车体与前轮
C5=(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
C6=-(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
%车体与后轮
C7=(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
C8=-(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
%路面与前轮
```

```
F1=zeros(ng,1);
F2=zeros(ng,1);
for i=1:ng
    F1(i)=f1t(xg(i));
    F2(i)=f2t(xg(i));
end
C9=uva(4,:)'-F1-ones(ng,1)*0.0508;
C10=-(uva(4,:)'-F1)-ones(ng,1)*0.0508;
%路面和后轮
C11=uva(5,:)'-F2-ones(ng,1)*0.0508;
C12=-(uva(5,:)'-F2)-ones(ng,1)*0.0508;

%组合约束
C=[C1;C2;C3;C4;C5;C6;C7;C8;C9;C10;C11;C12];
Ceq=[];


% 谱元法
function [xg,U_V_A]=fiveDegUVA(s,ne,inp,k1,k2,k3,c1,c2,c3)
%-------------------------------------------------------------%
% xg:全局点坐标                                               %
% U_V_A:是 20 乘(ne 乘 inp)矩阵，前五行、第二五行%
% 和最后五行分别是位移、速度和加速度                          %
% s 是时间段，ne：时间划分的单元个数                          %
% inp：每一个单元的插值个数                                   %
%-------------------------------------------------------------%
%------------
% input data
%------------
% s=2.035;   %时间的上界，单位是秒；下界是零
% s=2.0;
global gdm gdm2 bf9 bf10;
global ig xg;

% 中间节点密集型，节点必须是偶数
% ne=30;   % number of elements,stretch ratio
%--------------------------------
% 质量弹簧阻尼系统数据
%--------------------------------
```

```
M = sparse(5,5); B = sparse(5,5);
mm = [131.63  2042.54  472.70  43.85  43.85];
% mm=[1289.978 20016.9 4632.36 429.7 429.7];
% mm=[128.9978 2001.69 463.236 42.97 42.97];
% mm=[131.54  2042.16  472.37  43.82  43.82];

for i=1:5
   M(i,i)=mm(i);
end
B(4,2)=262688.976;B(4,3)=875.63;
B(5,4)=262688.976;B(5,5)=875.63;
L=3.048381;%车长
k4=262688.976;k5=262688.976;
c4=875.630;c5=875.630;

% k1=17512.6;k2=52537.8;k3=52537.8;
% c1=1751.26;c2=4378.15;c3=4378.15;

K0=[k1 -k1 -L*k1/12 0 0
   -k1 k1+k2+k3 L*(k1/4+k2-2*k3)/3 -k2 -k3
   -L*k1/12 L*(k1/4+k2-2*k3)/3 L^2*(k1/16+k2+4*k3)/9 -L*k2/3
   2*L*k3/3
   0 -k2 -L*k2/3 k2+k4 0
   0 -k3 2*L*k3/3 0 k3+k5];

C0=[c1 -c1 -L*c1/12 0 0
   -c1 c1+c2+c3 L*(c1/4+c2-2*c3)/3 -c2 -c3
   -L*c1/12 L*(c1/4+c2-2*c3)/3 L^2*(c1/16+c2+4*c3)/9 -L*c2/3
   2*L*c3/3
   0 -c2 -L*c2/3 c2+c4 0
   0 -c3 2*L*c3/3 0 c3+c5];
K1 = sparse(5,5); C1 = sparse(5,5);
K1(1:3,1:3) = K0(1:3,1:3); K1(2:3,4:5) = K0(2:3,4:5);
K1(4:5,2:3) = K0(4:5,2:3); K1(4,4) = K0(4,4); K1(5,5) = K0(5,5);
C1(1:3,1:3) = C0(1:3,1:3); C1(2:3,4:5) = C0(2:3,4:5);
C1(4:5,2:3) = C0(4:5,2:3); C1(4,4) = C0(4,4); C1(5,5) = C0(5,5);

K = M\K1;C = M\C1; B = M\B;
```

```
% K1=K;C1=C;

II = sparse(5,5);
for i = 1:5
   II(i,i) = 1;
end
Knr = size(K,1); Knc = size(K,2);
Cnr = size(C,1); Cnc = size(C,2);
As = sparse(5+Knr,Knc+Cnc);
%    zero=sparse(5);
% As=[zero -I
%    K C];
As(1:5,Knc+1:Knc+Cnc) = -II;
As(6:Knr+5,1:Knc) = K;
As(6:Knr+5,Knc+1:Knc+Cnc) = C;

ratio=1.0;
if ig==1
   for i=1:ne
      np(i)=inp;     % element polynomail order
   end

   %-------------------------------------
   % element node generation
   %-------------------------------------
   [xe,xen,xg,c,ng]=discr_s_1(0,s,ne,ratio,np);

   %计算各个单元的载荷
   for i=1:ne
      for j=1:np(i)+1
         f9(i,j)= k4/mm(4)*f1t(xen(i,j))+ c4/mm(4)*f1dt(xen
                  (i,j));
         f10(i,j)= k5/mm(5)*f2t(xen(i,j))+ c5/mm(5)*f2dt(xen
                   (i,j));
      end
   end

   %单元载荷组装
   bf9=F_sys_twoMSD_(ne,np,ng,c,f9);
```

```
    bf10=F_sys_twoMSD_(ne,np,ng,c,f10);
    %-----------------
    %element assembly
    %-----------------
    [gdm,gdm2]=KF_sys_twoMSD_(ne,xe,np,ng,c);
    %gdm(1,:)=0;gdm(:,1)=0;
    %gdm(1,1)=gdm(1,1)+1;
    gdm(1,1)=1;
    gdm2(1,1)=0;
    ig = ig +1;
end
%---------
% 全局组装
%---------
[row,colom]=size(As);
I=sparse(row);
for i = 1:row
    I(i,i) = 1;
end
GDM=kron(I,gdm);
GDM2=kron(As,gdm2);
Fcoef=-kron(I,gdm2);
%--------------
% linear solver
%--------------
[row,colom]=size(Fcoef);
F=zeros(row,1);
fL=length(bf9);

F(8*fL+1:9*fL)=bf9(:);
F(9*fL+1:10*fL)=bf10(:);

Fa=F;

F=Fcoef*F;
GDM=GDM-GDM2;

%边界条件处理
F(1)=0;
```

```
F(fL+1)=0;
F(2*fL+1)=0;
F(3*fL+1)=0;
F(4*fL+1)=0;
F(5*fL+1)=0;
F(6*fL+1)=0;
F(7*fL+1)=0;
F(8*fL+1)=0;
F(9*fL+1)=0;

%解线性方程组
% u=linsolve(GDM,F);
u = GDM\F;



%整理响应，求出加速度响应
uT=u';
FT=Fa';
uLen=length(uT);
uL10=uLen/10;
Uva=zeros(10,uL10);
Fva=zeros(10,uL10);
Uk=zeros(5,uL10);
Uc=zeros(5,uL10);
bF=zeros(5,uL10);

for i=1:10
    Uva(i,:)=uT((i-1)*uL10+1:i*uL10);
    Fva(i,:)=FT((i-1)*uL10+1:i*uL10);
end

%求出加速度
va=-As*Uva+Fva;

U_V_A=[Uva
    va];
return
```

```
% 区间离散子程序
function [xe,xen,xg,c,ng] = discr(x1,x2,ne,ratio,np)
%============================
% 离散区间 (x1, x2)为 "ne" 个单元,
% 生成 np 阶 Lobatto 插值节点
% xe:  单元节点
% xen: 单元插值节点
% xg:  全局节点
% c:   连接矩阵
% ng:  全局节点数
%============================

%--------------------
% 定义单元端点
%--------------------

xe = elm_line(x1,x2,ne,ratio);

%-------------------------
% 定义插值节点(xen)
%-------------------------

for l=1:ne

  mx = 0.5*(xe(l+1)+xe(l));
  dx = 0.5*(xe(l+1)-xe(l));

  xen(l,1) = xe(l);
     [zlob,wlob,P]=GLL (np(l));
     for je=1:np(l)-1
       xen(l,je+1) = mx + zlob(je+1)*dx;
     end

  xen(l,np(l)+1) = xe(l+1);
end
%---------------------------------
% 定义连接矩阵和全局节点
%---------------------------------
for l=1:ne
```

```
  Ic = Ic-1;
  for je=1:np(l)+1
    c(l,je) = Ic;
    xg(Ic) = xen(l,je);
  end
end

%--------------
% 节点总数
%--------------
ng = Ic-1;
return;

function km= KM_GLL(m)
%============================
% 计算单元刚度矩阵和单元质量矩阵
%============================

%-----------------------------------
% 生成谱节点和积分权系数
%-----------------------------------
[xi,w,P]=GLL(m);

%---------------------
% 计算微分矩阵
%---------------------

for i=1:m+1
  for j=1:m+1
     if(i ~= j)
       ndm(i,j) = 1.0/(xi(j)-xi(i));
       for k=1:m+1
         if(k ~= j)
            ndm(i,j) = ndm(i,j)*(xi(j)-xi(k));
         end
         if(k ~= i)
            ndm(i,j) = ndm(i,j)/(xi(i)-xi(k));
         end
       end
```

```
    end
  end
end

%-------------------------------------------------------------
% 用 Gauss-Lobatto 积分计算刚度矩阵和质量矩阵
%-------------------------------------------------------------

km = sparse(m+1,m+1);

for i=1:m+1    % 循环所有的节点
   km(i,:) = ndm(i,:).*w;
end

km(1,1)=km(1,1)+1;
km(m+1,m+1)=km(m+1,m+1)-1;
return;
```

2. 基于关键点法的动态响应程序

```
function [history,searchdir]=fiveDegMSD_OPT_ProblemOne_CP(arg1,
                              arg2)
%%
global CP_allData_No7colom;
history.x = [];
history.fval = [];
searchdir = [];

global ne inp;
global s ig;%gdm gdm2 bf
ig = 1;

s=2.0;%时间上限
% ne=10;inp=3;
ne = arg1;
inp = arg2;

% x00 前面三项分别是 k,c,b3(人工变量)
% k1=17512.6;k2=52537.8;k3=52537.8;
% c1=1751.26;c2=4378.15;c3=4378.15;
```

```
x00 = [17512.6 52537.8 52537.8 1751.26 4378.15 4378.15 8.448];
       % Starting guess at the solution
%x00 = [150; 0.02; 0.01];

lb=[8756.30 35025.20 35025.20 350.252 875.630 875.630 0.0254];
ub=[87563.0 175125.98 175125.98 8756.3 14010.08 14010.08 12.7];


opts=optimset('Algorithm','active-set','outputfcn',@outfun,'
              MaxFunEvals',5000,'MaxIter',500);
opts=optimset(opts,'TolFun',1.0e-12,'TolX',1.0e-12,'TolCon',
              1.0e-12,'Display','iter');
%opts=optimset(opts,'GradObj','on','GradConstr','on');
A=[];b=[];Aeq=[];beq=[];
t=cputime;
[x,fv,exitflag,ot] = fmincon(@fiveDegObj,x00,A,b,Aeq,beq,lb,ub,
                             @fiveDegConFun_CP,opts)
tcpu=cputime-t

% 优化结果写文件
fp = fopen(':\ \OptResultCriticalPoint_fivedeg_ProblemOne_CP.
           txt','a');
fprintf(fp,'****************************************\r\n');
fprintf(fp,'单元数 Nel=%d，插值点数 interp=%d\r\n',ne,inp);
fprintf(fp,'****************************************\r\n');
fprintf(fp,'%0.6f\r\n',x);
fprintf(fp,'iteration = %d\r\n',ot.iterations);
fprintf(fp,'t = %0.6f\r\n\r\n',tcpu);
fclose(fp);

if ot.funcCount<opts.MaxFunEvals && ot.iterations<opts.MaxIter
    disp('正常结束!');
    iter = ot.iterations;
    x3x=history.x;
    f=history.fval;
    xx=0:iter-1;
else
    disp('函数评估次数超过最大值!');
    iter = ot.iterations;
```

```
    x3x=history.x;
    f=history.fval;
    xx=0:iter;
end
%

%    subplot(2,2,1);

plot(xx,x3x(:,7),'-or','MarkerFaceColor','g');
% hold on;
% plot(xx,f,'-^m');
xlabel('迭代次数 n/次');ylabel('目标函数|y1|/m/s^2');box off;
set(gcf,'Color','w');
legendname = sprintf('GLL 点法: Nel=%d,m=%d',ne,inp); legend
             (legendname);
name = sprintf('E:\\CriticalPoint_OneDeg\\ProblemOne_FiveDeg_
              CP_obj_Nel%dm%d.fig',ne,inp);
saveas(gcf,name);close all;

CP_allData_No7colom = x3x;
matname = sprintf('E:\\CriticalPoint_OneDeg\\matfile\\fiveDeg_
                 CP_Nel%dm%d.mat',ne,inp);
save(matname,'CP_allData_No7colom');

    function stop = outfun(x,optimValues,state)
        stop = false;
        switch state
            case 'init'
                hold on
            case 'iter'
                % Concatenate current point and objective function
                % value with history. x must be a row vector.
                history.fval = [history.fval; optimValues.fval];
                history.x = [history.x; x];
                % Concatenate current search direction with
                % searchdir.
                searchdir = [searchdir;...
                    optimValues.searchdirection'];
                plot(x(1),x(2),'o','MarkerFaceColor','g');
```

```
                %Label points with iteration number and add title.
                text(x(1)+.015,x(2),...
                    num2str(optimValues.iteration));
                title('Sequence of Points Computed by fmincon');
            case 'done'
                hold off
            otherwise
        end
    end
end

function [C,Ceq] = fiveDegConfun_CP(x)
global s ne inp xen;
% global gdm gdm2;

k1 = x(1);k2 = x(2);k3 = x(3);
c1 = x(4);c2 = x(5);c3 = x(6);
d = x(7);

% 关键点法约束函数
[xen,xg,uva] = fiveDegUVA_CP(s,ne,inp,k1,k2,k3,c1,c2,c3);
ng = length(xg);
dd = d*ones(1,ng);
%加速度小于人工变量
% C1 = uva(16,:)'-dd';
% C2 = -uva(16,:)'-dd';
C10 = uva(16,:)';
C20 = -uva(16,:)';
[xen,C10_e] = yg2ye(xen,C10);
[xen,C20_e] = yg2ye(xen,C20);
[maxC10_e] = findCriticalPoint(xen,C10_e);
[maxC20_e] = findCriticalPoint(xen,C20_e);
C1 = maxC10_e - d;
C2 = maxC20_e - d;
%相对位移约束
L = 3.048381;%车长
%车体与司机
% C3 = uva(2,:)'+L/12*uva(3,:)'-uva(1,:)'-ones(ng,1)*0.0508;
% C4 = -(uva(2,:)'+L/12*uva(3,:)'-uva(1,:)')-ones(ng,1)*0.0508;
```

```
C30 = uva(2,:)'+L/12*uva(3,:)'-uva(1,:)';
C40 = -(uva(2,:)'+L/12*uva(3,:)'-uva(1,:)');
[xen,C30_e] = yg2ye(xen,C30);
[xen,C40_e] = yg2ye(xen,C40);
[maxC30_e] = findCriticalPoint(xen,C30_e);
[maxC40_e] = findCriticalPoint(xen,C40_e);
C3 = maxC30_e - 0.0508;
C4 = maxC40_e - 0.0508;
%车体与前轮
% C5 = (uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
% C6 = -(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
C50 = (uva(4,:)'-uva(2,:)'-L/3*uva(3,:)');
C60 = -(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)');
[xen,C50_e] = yg2ye(xen,C50);
[xen,C60_e] = yg2ye(xen,C60);
[maxC50_e] = findCriticalPoint(xen,C50_e);
[maxC60_e] = findCriticalPoint(xen,C60_e);
C5 = maxC50_e - 0.127;
C6 = maxC60_e - 0.127;
%车体与后轮
% C7 = (uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
% C8 = -(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
C70 = (uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)');
C80 = -(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)');
[xen,C70_e] = yg2ye(xen,C70);
[xen,C80_e] = yg2ye(xen,C80);
[maxC70_e] = findCriticalPoint(xen,C70_e);
[maxC80_e] = findCriticalPoint(xen,C80_e);
C7 = maxC70_e - 0.127;
C8 = maxC80_e - 0.127;
%路面与前轮
F1 = zeros(ng,1);
F2 = zeros(ng,1);
for i = 1:ng
    F1(i) = f1t(xg(i));
    F2(i) = f2t(xg(i));
end
% C9 = uva(4,:)'-F1-ones(ng,1)*0.0508;
% C10 = -(uva(4,:)'-F1)-ones(ng,1)*0.0508;
```

```
C90 = uva(4,:)'-F1;
C100 = -(uva(4,:)'-F1);
[xen,C90_e] = yg2ye(xen,C90);
[xen,C100_e] = yg2ye(xen,C100);
[maxC90_e] = findCriticalPoint(xen,C90_e);
[maxC100_e] = findCriticalPoint(xen,C100_e);
C9 = maxC90_e - 0.0508;
C10 = maxC100_e - 0.0508;
%路面和后轮
% C11 = uva(5,:)'-F2-ones(ng,1)*0.0508;
% C12 = -(uva(5,:)'-F2)-ones(ng,1)*0.0508;
C110 = uva(5,:)'-F2;
C120 = -(uva(5,:)'-F2);
[xen,C110_e] = yg2ye(xen,C110);
[xen,C120_e] = yg2ye(xen,C120);
[maxC110_e] = findCriticalPoint(xen,C110_e);
[maxC120_e] = findCriticalPoint(xen,C120_e);
C11 = maxC110_e - 0.0508;
C12 = maxC120_e - 0.0508;
%组合约束
C = [C1;C2;C3;C4;C5;C6;C7;C8;C9;C10;C11;C12];
Ceq = [];
% 寻找约束函数关键点
function [maxdisp] = findCriticalPoint(xen,disp_e)
[row,colom]=size(xen);

% opts=optimset('Algorithm','active-set','MaxFunEvals',3000');
% opts=optimset(opts,'TolFun',1.0e-15,'TolX',1.0e-15,'TolCon',
                1.0e-15,'Display','off');
for i = 1:row
    X = xen(i,:);
    Y = disp_e(i,:);
    t1=xen(i,1);t2=xen(i,colom);
    %[tzmax(i),zmax(i)] = fmincon(@Lagrange_CP,t0,[],[],[],[],
                          t1,t2,[],ops);
    [tmax(i),ymax(i)] = fminbnd(@(x) Lagrange_CP(x,X,Y),t1,t2);
end
maxdisp = ymax';
```

```
function Ys=Lagrange_CP(x,X,Y)
%拉格朗日插值,对给定的一组数据,求插值点的值
% global X Y;
Xs = x;
n=length(X);
nl=length(Xs);

for i=1:nl
    x=Xs(i);
    s=0;
    for j=1:n
        p=1;
        for k=1:n
            if k~=j
                p=p*(x-X(k))/(X(j)-X(k));
            end
        end
        s=s+p*Y(j);
    end
    Ys(i)=s;
end
```

附录 C　多元模型自适应与时间谱元法结合的动态优化

```
function [X, Y, Num_iter, Num_FE] = HAM_Kriging_RBF_MARS(range,
                                    iter,TOL)
%------------------------ Initialization --------------------
[mp, np] = size(range);
ex_points = rlhsamp(8, np, range);

% 在采样点中加入所有顶点
vertex = compvertex(range);
ex_points = [ex_points;vertex];

assignin('base','vertex',vertex);

% 处理仿真次数以节约时间
assignin('base','ig',1);

[ex_points_value,num_class,st] = calobj(ex_points);

% 输出目标函数的名称及其他
obj_name(num_class);
% 约束处理：构造三个元模型，将不满足约束的点删除
% points = ex_points; points_value = ex_points_value;
[ex_points,ex_points_value] = handle_ST(ex_points,ex_points_
                                        value,st);

global_min = min(ex_points_value);



% --------------Parameters for Kriging Metamodel--------------
……
```

该部分请参阅顾继超论文（在本书中，类似的标注是为了避免侵权，因为其中部分程序是顾继超编写的。下同）

```
% ------------Parameters for MARS Metamodel-------------------
params = aresparams(121, 1, [], [], [], 3);

% -----------------------------------------------------------
iteration = 1;
criteria = 1;
Tol = TOL;

% mhp 2012.7.26
minimum_seriation = [];
%

% ------------Main Algorithm Loop-----------------------------
while(criteria > Tol)

    %     -------------------Construct the Metamodels----------
    quacoef = aresbuild(ex_points, ex_points_value, params); %
              fit MARS model
    ......
```

该部分请参阅顾继超论文。

```
        %          ------------Kriging and MARS----------------
        both_index_kq = [];
        both_index_qk = [];
        both_points_kq = [];
        both_points_qk = [];
        points_kq = [];
        for i = 1 : 100
            for j = 1 : 100
                if(max(abs(t_points_k(i, :) - t_points_q(j, :)))<
                   eps)
                    both_index_kq(i) = i;
                    both_index_qk(j) = j;
                end
            end
        end
        if(any(both_index_kq))
            both_index_kq(both_index_kq == 0) = [];
```

```
            both_index_qk(both_index_qk == 0)  = [];
            both_points_kq = t_points_k(both_index_kq, :);
            both_points_qk = t_points_q(both_index_qk, :);
            points_kq = both_points_kq;
        end
        %          -----------RBF and MARS-------------------
        both_index_qr = [];
        both_index_rq = [];
        both_points_qr = [];
        both_points_rq = [];
        points_qr = [];
        for i = 1 : 100
            for j = 1 : 100
                if(max(abs(t_points_q(i, :) - t_points_r(j, :)))<
                    eps)
                    both_index_qr(i) = i;
                    both_index_rq(j) = j;
                end
            end
        end
        if(any(both_index_qr))
            both_index_qr(both_index_qr == 0) = [];
            both_index_rq(both_index_rq == 0)  = [];
            both_points_qr = t_points_k(both_index_qr, :);
            both_points_rq = t_points_r(both_index_rq, :);
            points_qr = both_points_qr;
        end
    else
        %          ---------------- Kriging and RBF-------------
        ……
```

该部分请参阅顾继超论文。

```
        %          ------------Kriging and MARS----------------
        both_index_kq = [];
        both_index_qk = [];
        both_points_kq = [];
        both_points_qk = [];
        points_kq = [];
        for i = 1 : 100
            for j = 1 : 100
```

```
            if(max(abs(t_points_q(i, :) - t_points_k(j, :)))<
               eps)
                both_index_qk(i) = i;
                both_index_kq(j) = j;
            end
        end
    end
    if(any(both_index_qk))
        both_index_kq(both_index_kq == 0) = [];
        both_index_qk(both_index_qk == 0)  = [];
        both_points_kq = t_points_k(both_index_kq, :);
        both_points_qk = t_points_q(both_index_qk, :);
        points_kq = both_points_kq;
    end
    %          -----------RBF and MARS-------------------
    both_index_qr = [];
    both_index_rq = [];
    both_points_qr = [];
    both_points_rq = [];
    points_qr = [];
    for i = 1 : 100
        for j = 1 : 100
            if(max(abs(t_points_r(i, :) - t_points_q(j, :)))<
            eps)
                both_index_rq(i) = i;
                both_index_qr(j) = j;
            end
        end
    end
    if(any(both_index_qr))
        both_index_qr(both_index_qr == 0) = [];
        both_index_rq(both_index_rq == 0)  = [];
        both_points_qr = t_points_q(both_index_qr, :);
        both_points_rq = t_points_r(both_index_rq, :);
        points_qr = both_points_qr;
    end
end

%      -----------Among the Three Metamodels---------------
```

……

该部分请参阅顾继超论文。

```
new_ex_points = [ex_points_k; ex_points_r; ex_points_q;
                 ex_points_kq;...
    ex_points_qr; ex_points_kr; ex_points_kqr];

%   -------------Check the duplicate points---------------
[mre nre] = size(new_ex_points);
eorder = zeros(1);
for i = 1 : mre
    for j = i+1 : mre
        if (max(abs(new_ex_points(i, :)- new_ex_points(j,:)))
            < eps)
            eorder(i) = i;
        end
    end
end
if (any(eorder))
    eorder(eorder == 0) =[];
    new_ex_points(eorder, :) = [];
end

[mrn nrn] = size(new_ex_points);
[mre nre] = size(ex_points);
eorder = zeros(1);
for i = 1 : mrn
    for j = 1 : mre
        if (max(abs(new_ex_points(i, :) - ex_points(j, :)))
            < eps)
            eorder(i) = i;
        end
    end
end
if (any(eorder))
    eorder(eorder == 0) =[];
    new_ex_points(eorder, :) = [];
end

%     ------------------ Termination Criteria-------------
```

```
    % new_ex_points_value = calobj(new_ex_points);

    [new_ex_points_value,num_class,st] = calobj(new_ex_points);
    [new_ex_points,new_ex_points_value] = handle_ST(new_ex_
                                          points,new_ex_poi
                                          nts_value,st);
    ……
```

该部分请参阅顾继超论文。

```
assignin('base','minimum_seriation',minimum_seriation);
plot(1:size(minimum_seriation,1),minimum_seriation);
assignin('base','ex_points',ex_points);
assignin('base','ex_points_value',ex_points_value);


function S = rlhsamp(m, n, range)
%LHSAMP Latin hypercube distributed random numbers
%
% Call:    S = lhsamp
%       S = lhsamp(m)
%       S = lhsamp(m, n)
%
% range :  2*n matrix with lower and upper limits
% m : number of sample points to generate,if unspecified m = 1
% n : number of dimensions, if unspecified n = m
%
% S : the generated n dimensional m sample points chosen from
%    uniform distributions on m subdivions any interval

……
```

该部分请参阅顾继超论文。

```
function [points,points_value] = handle_ST(ex_points,ex_points_
                                  value,st)
% 取出约束函数需要的值，删除不满足约束的点及其对应的函数值
[m,n] = size(st);
no_st = [];
for i = 1:m
   for j = 1:n
```

```
        if st(i,j) > 0
            no_st = [no_st;i];
            break;
        end
    end
end

ex_points(no_st,:) = [];
ex_points_value(no_st,:) = [];

points = ex_points;
points_value = ex_points_value;
m2 = size(points,1);
msg = sprintf('原始点数为%d，约束以后点数为%d',m,m2);
disp(msg);

function [Y,num_class,st] = calobj(x)
%CAL_FUNC  Calculate the objective function value,
% Call:    Y = calfunc(x)
% x : input variable
% Y : the function value
% % % %
% [y,num_class] = [m n] = size(x);
for i = 1 : m-1
    [Y(i,:),nm_class,st(i,:)] = obj(x(i,:));
end
[Y(m,:) num_class,st(m,:)] = obj(x(m,:));

function [y,num_class,st] = obj(x)
%% %%%%%%%%%%%%%%%%%%%%%%%%
% 五自由度汽车悬架系统优化设计（问题一）
% %%%%%%%%%%%%%%%%%%%%%%%%%%
% lb=[8756.30 35025.20 35025.20 350.252 875.630 875.630];
% ub=[87563.0 175125.98 175125.98 8756.3 14010.08 14010.08];

% num_class = 23;
% [uva,xg] = FiveDOFautomobileSim(x);
% A = uva(16,:);
% Amax = max(abs(A));
```

```
% y = Amax;
%
% ng=length(xg);
% %相对位移约束
% L=3.048381;%车长
% %车体与司机
% C3=uva(2,:)'+L/12*uva(3,:)'-uva(1,:)'-ones(ng,1)*0.0508;
%C4=-(uva(2,:)'+L/12*uva(3,:)'-uva(1,:)')-ones(ng,1)*0.0508;
% C1 = max([max(C3) max(C4)]);
%
% %车体与前轮
% C5=(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
% C6=-(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
% C2 = max([max(C5) max(C6)]);
%
% %车体与后轮
% C7=(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
%C8=-(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
% C3 = max([max(C7) max(C8)]);
%
% %路面与前轮
% F1=zeros(ng,1);
% F2=zeros(ng,1);
% for i=1:ng
%     F1(i)=f1t(xg(i));
%     F2(i)=f2t(xg(i));
% end
% C9=uva(4,:)'-F1-ones(ng,1)*0.0508;
% C10=-(uva(4,:)'-F1)-ones(ng,1)*0.0508;
% C4 = max([max(C9) max(C10)]);
%
% %路面和后轮
% C11=uva(5,:)'-F2-ones(ng,1)*0.0508;
% C12=-(uva(5,:)'-F2)-ones(ng,1)*0.0508;
% C5 = max([max(C11) max(C12)]);
% st = [C1 C2 C3 C4 C5];
%% %%%%%%%%%%%%%%%%%%%%%
% 五自由度汽车悬架系统优化设计（问题二）
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% lb=[8756.30 35025.20 35025.20 350.252 875.630 875.630];
% ub=[87563.0 175125.98 175125.98 8756.3 14010.08 14010.08];

num_class = 25;
[uva,xg] = Automobile_5DOF_Problem2_Sim(x);
A = uva(16,:);
Amax = max(abs(A));
y = Amax;

ng=length(xg);
L=3.048381;%车长
%车体与司机
C3=uva(2,:)'+L/12*uva(3,:)'-uva(1,:)'-ones(ng,1)*0.0508;
C4=-(uva(2,:)'+L/12*uva(3,:)'-uva(1,:)')-ones(ng,1)*0.0508;
C1 = max([max(C3) max(C4)]);
%车体与前轮
C5=(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
C6=-(uva(4,:)'-uva(2,:)'-L/3*uva(3,:)')-ones(ng,1)*0.127;
C2 = max([max(C5) max(C6)]);
%车体与后轮
C7=(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
C8=-(uva(5,:)'-uva(2,:)'+2*L/3*uva(3,:)')-ones(ng,1)*0.127;
C3 = max([max(C7) max(C8)]);
%路面与前轮
F1=zeros(ng,1);
F2=zeros(ng,1);
for i=1:ng
    F1(i)=f1t(xg(i));
    F2(i)=f2t(xg(i));
end
C9=uva(4,:)'-F1-ones(ng,1)*0.0508;
C10=-(uva(4,:)'-F1)-ones(ng,1)*0.0508;
C4 = max([max(C9) max(C10)]);
%路面和后轮
C11=uva(5,:)'-F2-ones(ng,1)*0.0508;
C12=-(uva(5,:)'-F2)-ones(ng,1)*0.0508;
C5 = max([max(C11) max(C12)]);
st = [C1 C2 C3 C4 C5];
```

```
function [U_V_A,xg]=Automobile_5DOF_Problem2_Sim(x)
%------------------------------------------------------------%
% xg:全局点坐标                                              %
% U_V_A:是20乘(ne乘inp)矩阵，前五行、第二五行%
% 和最后五行分别是位移、速度和加速度                    %
% s是时间段，ne：时间划分的单元个数                     %
% inp：每一个单元的插值个数                                  %
%------------------------------------------------------------%
%------------
% input data
%------------
% s=2.035;   %时间的上界，单位是秒；下界是零
% s=2.0;
% global gdm gdm2 bf9 bf10;
% global ig xg;
[xg,U_V_A]=fiveDegUVA(s,ne,inp,x)。
```

附录 D　MARS 的动态响应优化算法

```
! 调动 ANSYS 仿真器进行优化悬臂梁外形形状的 APDL 代码
/CLEAR
*DIM,EXA,,1,4
*VREAD,EXA(1,1),D:\DUQU,TXT,,JIK,4,1
 (4F15.8)

THK11=EXA(1,1)
THK12=XAE(1,
THK13=XAE(1,
THK14=XAE(1,4)
*CFOPEN,D:\LOOKDUQU,TXT
*VWRITE,XAE(1,1)
 (F6.8)
*CFCLOS

/PREP7
SMRT,OFF

ANTYPE,STATIC
ET,1,PLANE42

MP,EX,1,68952
MP,NUXY,1,0.3

K,1
K,5,254
KFILL
K,6,,THK11
K,7,63.5,THK12
K,8,127,THK13
K,9,190.5,THK14
K,10,254,3.81
```

```
SPLINE,6,7,8,9,10
L,1,6
*REPEAT,5,1,1
LSEL,S,LINE,,5,9
LESIZE,ALL,,,1
LSEL,ALL
A,1,2,7,6
*REPEAT,4,1,1,1,1
ESIZE,,4
AMESH,ALL

NSEL,S,LOC,Y
DSYM,SYMM,X
NSEL,S,LOC,X
DSYM,ASYM,Y
NSEL,ALL
DK,1,ALL,0


/SOLU
SOLVE
FINISH

/POST1
SET,LAST
ETABLE,VOLU,VOLU
PRNSOL,S,PRIN
NSORT,S,1
NSEL,S,LOC,X,0,228.6
*GET,STRS,SORT,,MAX
NSEL,ALL
SSUM
*GET,TVOL,SSUM,,ITEM,VOLU
TVOL=TVOL*2
NSEL,S,LOC,X,253,255
PRNSOL,U,Y
NSORT,U,Y,,1
PRNSOL,U,Y
*GET,DEFL,SORT,,MAX
```

```
*STATUS,PARM
DEFL=ABS(DEFL)


DIIFF1=THK11-THK12
DIIFF2=THK12-THK13
DIIFF3=THK13-THK14

*DIM,DIS,,6,1
DIISS(1,1)=TVOL
DIISS(2,1)=DEFL
DIISS(3,1)=DIIFF1
DIISS(4,1)=DIIFF2
DIISS(5,1)=DIIFF3
DIISS(6,1)=STRS
*CFOPEN,D:\OUTPUT,TXT

*VWRITE,DIISS(1,1)
(F15.8)
*CFCLOS
FINISH
```

```
% MARS 与移动极限相结合主程序
global  model1 model2 model3 model4 model5 model6 model7 model8
model9;
MaxOuterIterNumber = 55; % 最大迭代次数
InitialMoveLimit = 0.25; % 移动极限初始值
RelativeCovergenceDelta = 1.0e-6; % 相对收敛准则
SubjectRelativeBeyow = 1e-5; % 约束相对违约量
MeanY5 = 1.0;
MeanCovergenceDelta = 1.0e-6;
ML = 0.025; % Move Limits
LB = 0.25; % TRR Lower Limit
UB = 0.75; % TRR Upper Limit
r    = 0.10;   % ML Reduction Factor
e    = 1.25;   % ML Expansion Factor
isUpdateModel = 1;
% first Iterative
m = 16; n = 2;
```

```
range = [0.05*ones(1,2); 0.5*ones(1,2)];

load XYT_PlaneTruss.mat;
x0 = XT;
y  = YT;

% ³初始值
xx0 = ones(n,1)*0.2;
xfirst = xx0;

% 构造 MARS 近似模型
params = aresparams(200, [], false, [], [], 20);
model1 = aresbuild(XT, YT(:,1), params,[],[],false);
model2 = aresbuild(XT, YT(:,2), params,[],[],false);
model3 = aresbuild(XT, YT(:,3), params,[],[],false);
model4 = aresbuild(XT, YT(:,4), params,[],[],false);
model5 = aresbuild(XT, YT(:,5), params,[],[],false);
model6 = aresbuild(XT, YT(:,6), params,[],[],false);
model7 = aresbuild(XT, YT(:,7), params,[],[],false);
model8 = aresbuild(XT, YT(:,8), params,[],[],false);
model9 = aresbuild(XT, YT(:,9), params,[],[],false);

% 初始值对应的精确值
F1 = GetAnsysValue_PlaneTruss(xx0); %
% 初始值对应的近似值
A1(1) = arespredict(model1,xx0');
A1(2) = arespredict(model2,xx0');
A1(3) = arespredict(model3,xx0');
A1(4) = arespredict(model4,xx0');
A1(5) = arespredict(model5,xx0');
A1(6) = arespredict(model6,xx0');
A1(7) = arespredict(model7,xx0');
A1(8) = arespredict(model8,xx0');
A1(9) = arespredict(model9,xx0');

% 涉及变量的上下界限
initlb = 0.05*ones(2,1);
initub = 0.5*ones(2,1);
```

```
lb = xx0*(1-InitialMoveLimit); % 20×1
ub = xx0*(1+InitialMoveLimit);% 20×1
% 判断当前取值是否越界
for i = 1:n
    if(lb(i) < initlb(i))
        lb(i) = initlb(i);
    end
    if(ub(i) > initub(i))
        ub(i) = initub(i);
    end
end

opts = optimset('Algorithm','sqp');
problem = createOptimProblem('fmincon','objective',@Ansys_
          Obj,...
    'x0',xx0,'lb',lb,'ub',ub,'nonlcon',@AnsysCon,'options',
    opts);
gs = GlobalSearch;
[x,fval] = run(gs,problem);
y = GetAnsysValue_PlaneTruss(x);
msg = sprintf(' Iteration       Objective      MaxConstraint');
              disp(msg);
msg = sprintf('     %d       %d       %d',1,y(1),max(y(2:end)));
              disp(msg);

xforward = x;
xsecond = x;

% 第一次迭代结束，最优点对应的目标函数和约束函数精确值
F2 = GetAnsysValue_PlaneTruss(x); %
%  第一次迭代结束，最优点对应的目标函数和约束函数的近似值
A2(1) = arespredict(model1,x');
A2(2) = arespredict(model2,x');
A2(3) = arespredict(model3,x');
A2(4) = arespredict(model4,x');
A2(5) = arespredict(model5,x');
A2(6) = arespredict(model6,x');
A2(7) = arespredict(model7,xx0');
A2(8) = arespredict(model8,xx0');
```

```
A2(9) = arespredict(model9,xx0');

% 计算 TRR
TRR = (F1-F2)./(A1-A2);

% 取出目标函数值
TRR = TRR(1);
plotTRR(1) = TRR;

Xper = x;
Yper = F2(1);
plotY(2) = F2(1);

% 更新总数据库
XT = cat(1,XT,x');
YT = cat(1,YT,F2);


for IterNum = 1:MaxOuterIterNumber
    %
    IterativeNumber = IterNum;

    xx0 = x;
    model1 = aresbuild(XT, YT(:,1), params,[],[],false);
    model2 = aresbuild(XT, YT(:,2), params,[],[],false);
    model3 = aresbuild(XT, YT(:,3), params,[],[],false);
    model4 = aresbuild(XT, YT(:,4), params,[],[],false);
    model5 = aresbuild(XT, YT(:,5), params,[],[],false);
    model6 = aresbuild(XT, YT(:,6), params,[],[],false);
    model7 = aresbuild(XT, YT(:,7), params,[],[],false);
    model8 = aresbuild(XT, YT(:,8), params,[],[],false);
    model9 = aresbuild(XT, YT(:,9), params,[],[],false);
    %
    TRR = (F1-F2)./(A1-A2);
    %
    TRR = min(TRR);
    plotTRR(IterNum+1) = TRR;
    %
    F1 = F2;
```

```
A1 = A2;

if(IterNum == 1 || IterNum == 2)
    subbey = max(YT);
else
    [rw,ar] = size(YT);
    subjectbeyow = YT(rw,2:9);
    subbey = max(subjectbeyow);
end
if(subbey > SubjectRelativeBeyow)
    xx0 = xforward;
    ML = ML*0.95;
elseif(subbey < -SubjectRelativeBeyow)
    ML = ML*1.05;
end

if(IterNum == 1)
    % 选择 Move Limit 步长
    if(TRR < LB)
        ML = ML*r;
        xx0 = xforward;
        isUpdateModel = 0;
    elseif(TRR > UB || isnan(TRR) == 1)
        ML = ML*e;
        isupdate = 1;
    else
        ML = ML;
        isupdate = 1;
    end
    plotML(IterNum) = ML;

    if(ML < 0.0005)
        ML = 0.02;
    end
else
    if(one == 0)
        if(fifth == 0)
            %break;
        elseif(fifth == 1)
```

```
            if(forth ==0)
                ML = 1/1.5;
            elseif(forth == 1)
                ML = 1/1.05;
            end
        end
        xx0 = xforward; %
        isUpdateModel = 1;
    elseif(one == 1)
        if(second == 1)
            if(fifth == 0)
                %break;
            elseif(fifth == 1)
                if(sixth == 0)
                    ML = 1/4;
                elseif(sixth == 1)
                    ML = 1/2;
                end
            end
        elseif(second == 0)
            if(third == 0)
                ML = 1/1.5;
            elseif(third == 1)
                if(forth == 0)
                    ML = 1;
                elseif(forth == 1)
                    ML = 1/0.8;
                end
            end
        end
        isUpdateModel = 1;
    elseif(one == 2)
        if(second == 1)
            if(fifth == 0)
                %break;
            elseif(fifth == 1)
                if(sixth == 0)
                    ML = 1/4;
                elseif(sixth == 1)
```

```
                ML = 1/2;
            end
        end
        isUpdateModel = 1;
    elseif(sixth == 0)
        if(third == 0)
            ML = 1/1.5;
        elseif(third == 1)
            if(forth == 0)
                ML = 1;
            elseif(forth == 1)
                ML = 1/0.8;
            end
        end
        isUpdateModel = 0;
        xx0 = xforward;
    end
end
end
if(IterNum ~= 1)
    plotML(IterNum) = ML;
end

xforward = x;
lb = xx0*(1-ML);
ub = xx0*(1+ML);
%
if(min(lb-initlb)<0)
    lb = initlb;
end
if(min(initub-ub)<0)
    ub = initub;
end
% 判断是否超界
for i = 1:n
    if(lb(i) < initlb(i))
        lb(i) = initlb(i);
    end
    if(ub(i) > initub(i))
```

```
        ub(i) = initub(i);
    end
end

problem = createOptimProblem('fmincon','objective',@Ansys_
          Obj,...
    'x0',xx0,'lb',lb,'ub',ub,'nonlcon',@AnsysCon,'options',
    opts);
gs = GlobalSearch;
[x,fval] = run(gs,problem);
y = GetAnsysValue_PlaneTruss(x);
msg = sprintf('    %d         %d           %d',IterNum+1,
       y(1),max(y(2:end))); disp(msg);
% 更新数据库
if(isUpdateModel == 1)
    XT = cat(1,XT,x');
    YT = cat(1,YT,y);
end
% 判断是否收敛
errY = abs((y(1)-Yper)/y(1));
errX = norm(x-Xper);
if errY<=RelativeCovergenceDelta||errX<RelativeCovergenceDelta
    break;
end
Xper = x;
Yper = y(1);
%
F2 = GetAnsysValue_PlaneTruss(x); %
%
A2(1) = arespredict(model1,x');
A2(2) = arespredict(model2,x');
A2(3) = arespredict(model3,x');
A2(4) = arespredict(model4,x');
A2(5) = arespredict(model5,x');
A2(6) = arespredict(model6,x');
A2(7) = arespredict(model7,xx0');
A2(8) = arespredict(model8,xx0');
A2(9) = arespredict(model9,xx0');
```

```
plotY(IterNum+2) = F2(1);

if IterNum >= 4
   MeanY5 = (plotY(end-4:end) - plotY(end-5:end-1))/5;
end
if MeanY5 <= MeanCovergenceDelta
   break;
end

xthird = x;

% The first indicator is the quality index of the approximation
functions.
GF = max(abs(F2(2:9)));
GA = max(abs(A2(2:9)));
E = abs((GA-GF)/GF);
S = max((ub-lb)./(initub-initlb));
if(E>0.25*S)
   one = 0; % bad
elseif(E<0.01*S)
   one = 2; % good
else
   one = 1; % reasonable
end
% The second indicator is the location of the sub-optimum point
in the
% current search subregion. When none of the current move
limits is active,
% the solution is considered as "internal", otherwise it is
denoted as
% "external".
SubOptPointLocationU = max(ub-x);
SubOptPointLocationL = max(x-lb);
if(SubOptPointLocationU <= 0.001 || SubOptPointLocationL <=
                            0.001)
   second = 0; % external
else
   second = 1; % internal
```

```
end
% A third and fourth indicator are based on the movement
history. For that
% purpose the angle between the last two move vectors is
measured by  theta
theta  =  dot(xthird-xsecond,xsecond-xfirst)/(norm(xthird-
             xsecond))/norm(xsecond-xfirst);
if(theta <= 0)
   third = 0; % backward
else
   third = 1; % forward
   if(theta <= 0.5)
      forth = 0; % curved
   else
      forth = 1; % straight
   end
end

xfirst = xsecond;
xsecond = xthird;
% The fifth indicator, used in the termination criterion, is
the size of
% the present search subregion, it is denoted as "small"
or "large".
if(one == 0 && S < 0.005)
   fifth = 0; % small
elseif((one == 1 || one == 2 ) && S < 0.1)
   fifth = 0;
else
   fifth = 1; % large
end
% The sixth indicator is based on the most active constraint.
FF2 = F2(1:5);
FF2 = max(FF2);
if(FF2<0.1 && FF2>-0.1)
   sixth = 0; % close
else
   sixth = 1; % far
end
```

```
end

% 绘制目标趋势图
plotYNum = length(plotY);
px = 1:plotYNum;
plot(px,plotY,'o-r','MarkerFaceColor','g');
box off; set(gcf,'Color','w');
```

附录 E　基于模糊聚类的全局动态响应优化算法

```
% Michalewicz function = [0 0;pi pi];
% range = [0 0;pi pi];

% Himmelblau function [-6 6]
% range = [-6 -6;6 6];

% Golden-Price function (GP) with n = 2: x = [-2 -2; 2 2]
% range = [-2 -2;2 2];

%six-hump camel-back (SC) function  x = [-2 -2; 2 2]
% range = [-2 -2;2 2];

%Leon function[-10 10]
range = [-10 -10;10 10];

% 首先，采样小样本点
initnum = 24;

np = size(range,2);
Xsm = rlhsamp(initnum,size(range,2),range);
Ysm = calobj(Xsm);

% 寻找聚类中心
ccx = findccenter(Xsm,Ysm,range);
% 检查是否有重复点
[dupx,dupy,dedx] = checkduplicatepoints(ccx,Xsm,Ysm);

% 处理重复点
[Xsm,Ysm,ccx,ccy] = handleduplicatepoints(Xsm,Ysm,dupx,dupy,
                    dedx);
```

```
[Yo,Yoi] = sort(Ysm);
global_value = Yo(1);
global_points = Xsm(Yoi(1),:);
msg = sprintf('The 1th iterative global min is %d',global_value);
disp(msg);

criteria = false;
iter = 1;
MaxIter = 30;
num = 8;
locnum = 16;
Tol = 0.01;
criteria = 1;
while(criteria > Tol)
    % 构造重要区域
    middle = sort(ccx);
    middle_range = [middle(1, :); middle(end, :)];

    if(rem(iter,4) == 0 || iter == 1)
        tempXsm = rlhsamp(locnum,np,middle_range);
        [tempXsm] = checkduplicatepoints2(tempXsm,Xsm);
        tempYsm = calobj(tempXsm);
        Xsm = [Xsm;tempXsm];
        Ysm = [Ysm;tempYsm];
        ccx = findccenter(Xsm,Ysm,middle_range);
        % 检查是否有重复点
        [dupx,dupy,dedx] = checkduplicatepoints(ccx,Xsm,Ysm);
        % 处理重复点
        [Xsm,Ysm,ccx,ccy] = handleduplicatepoints(Xsm,Ysm,dupx,
                            dupy,dedx);
        % Hooke Jeeves 局部搜索
        [Xloc,Yloc] = findpromisingrangexy(Xsm,Ysm,middle_range);

        [xloc] = localsearch(Xloc,Yloc,middle_range);
        yloc = calobj(xloc);
        msg = sprintf('Local search result is %d.',yloc);
        disp(msg);
        % 检查是否有重复点
        [dupx,dupy,dedx] = checkduplicatepoints(xloc,Xsm,Ysm);
```

```
        % 处理重复点
        [Xsm,Ysm,ccx0,ccy0] = handleduplicatepoints(Xsm,Ysm,dupx,
                              dupy,dedx);
    else
        tempXsm = rlhsamp(num,np,range);
        [tempXsm] = checkduplicatepoints2(tempXsm,Xsm);
        if(~isempty(tempXsm))
            tempYsm = calobj(tempXsm);
            Xsm = [Xsm;tempXsm];
            Ysm = [Ysm;tempYsm];
            ccx = findccenter(Xsm,Ysm,range);
            % 检查是否有重复点
            [dupx,dupy,dedx]=checkduplicatepoints(ccx,Xsm,Ysm);
            % 处理重复点
            [Xsm,Ysm,ccx,ccy] = handleduplicatepoints(Xsm,Ysm,
                                dupx,dupy,dedx);
        end
    end

    % 计算聚类中心的几何中心
    gcx = sum(ccx)/(size(ccx,1)+1);
    msg = sprintf('Geometric Center is %d.',calobj(gcx));
    disp(msg);
    %gcy = calobj(gcx);
    [dupx,dupy,dedx] = checkduplicatepoints(gcx,Xsm,Ysm);
    if(isempty(dupx))
        Xsm = [Xsm;gcx];
        Ysm = [Ysm;calobj(gcx)];
    end
%     % 处理重复点
%     [Xsm,Ysm,ccx,ccy] = handleduplicatepoints(Xsm,Ysm,dupx,
                        dupy,dedx);

    [Yo,Yoi] = sort(Ysm);
    global_value = [global_value;Yo(1)];
    global_points = [global_points;Xsm(Yoi(1),:)];
    [go,goi] = sort(global_value);

    [dis_global, order_global] = sort(Ysm);
```

```
    mean_ex_value(iter) = mean(dis_global(1 : 5));
    if(iter > 3)
        if(abs(mean_ex_value(end) - mean_ex_value(end - 1)) > 0)
            criteria = abs(mean_ex_value(end) - mean_ex_value
                       (end - 1));
        end
    end
    if(iter > MaxIter)
        criteria = eps;
    end
    iter = iter + 1;
    msg = sprintf('The %dth iterative global min is %d',iter,
                  go(1));
    disp(msg);

end
plot(1:length(global_value),global_value','o-','MarkerFaceCo
     lor','r');

% 寻找聚类中心
function cc = findccenter(Xsm,Ysm,range)
% Banana Functionrange = [-2 -2;2 2];
% initnum = 64;
% range = [0 0;pi pi];
% Xsm = rlhsamp(initnum,size(range,2),range);
% Ysm = calobj(Xsm);

np = size(range,2);
% ---------------Parameters for Kriging Metamodel-------------
theta = [10 ];
lob = [1e-1  ];
upb = [20  ];
theta = repmat(theta, 1, np) ;
lob = repmat(lob, 1, np);
upb =  repmat(upb, 1, np);
% ------------------------------------------------------------

[dmodel, perf] = dacefit(Xsm, Ysm, @regpoly0,...
    @corrgauss, theta, lob, upb);  % fit  kriging metamodel
```

```
cheapnum = 10000;
XS = rlhsamp(cheapnum,np,range);

Yc = [];
for i =  1: cheapnum
   Yc(i,:) = predictor(XS(i,:), dmodel);%Decrease the use of
             memory
end

[Ycso,Yciso] = sort(Yc);
XS100 = XS(Yciso(1:cheapnum/100),:);

cc = myFCMcall(XS100,3,false);

% [x1,x2] = meshgrid(0:0.01:pi,0:0.01:pi);
% % y = 100 * (x2 - x1.^2).^2 + (1 - x1).^2;
% % y = 100 * (x2 - x1.^3).^2 + (x1 - 1).^2;
% y = [];
%
% for i = 1:size(x1,1)
%    for j = 1:size(x1,2)
%       x12 = [x1(i,j),x2(i,j)];
%       y(i,j) = mich(x12);
%    end
% end
% figure;
% contour(x1,x2,y);
% hold on;
% plot(cc(:,1),cc(:,2),'o','MarkerFaceColor','m');

% 寻找聚类中心子函数
function [ccenter] = myFCMcall(XS,c,verbose)
data.X = XS;
%parameters
param.c=c;
param.m=2;
param.e=1e-6;
param.ro=ones(1,param.c);
```

```
param.val=1;
%normalization
data=clust_normalize(data,'range');
%clustering
result = FCMclust(data,param);
clustercenter = result.cluster.v;
xmin = repmat(data.min,[param.c 1]);
xmax = repmat(data.max,[param.c 1]);

ccenter = clustercenter.*(xmax-xmin) + xmin;

if(verbose)
    data = clust_denormalize(data);
    plot(data.X(:,1),data.X(:,2),'b.',ccenter(:,1),ccenter(:,
    2),'ro');
    hold on
    %draw contour-map
    new.X=data.X;
    result.cluster.v = ccenter;
    eval=clusteval(new,result,param);
    %validation
    result = validity(result,data,param);
end

% 检查是否有重复点
function [dupx,dupy,dedx] = checkduplicatepoints(new_ex_points,
                            ex_points,ex_points_value)
dedx = new_ex_points;
[mrn nrn] = size(new_ex_points);
[mre nre] = size(ex_points);
eorder = zeros(1);
exorder = zeros(1);
dupx = [];
dupy = [];
for i = 1 : mrn
    for j = 1 : mre
        if(max(abs(new_ex_points(i, :) - ex_points(j, :))) < eps)
            eorder(i) = i;
            exorder(j) = j;
```

```
        end
    end
end

if (any(eorder))
    eorder(eorder == 0) =[];
    exorder(exorder == 0) = [];
    dedx(eorder, :) = [];
    dupx = ex_points(exorder,:);
    dupy = ex_points_value(exorder');
end

% 处理重复点
function[Xsm,Ysm,ccx,ccy]=handleduplicatepoints(Xsm,Ysm,dupx,
                                                dupy,dedx)
if(size(dupx,1) == 0)
    Xsm = [Xsm;dedx];
    dedy = calobj(dedx);
    Ysm = [Ysm;dedy];
    ccy = dedy;
    ccx = dedx;
elseif(size(dedx,1) == 0)
    ccy = dupy;
    ccx = dupx;
else
    dedy = calobj(dedx);
    ccx = [dupx;dedx];
    ccy = [dupy;dedy];
    Xsm = [Xsm;dedx];
    Ysm = [Ysm;dedy];
end

% Hooke Jeeves 局部搜索
function [Xloc,Yloc] = findpromisingrangexy(X,Y,range)
[m,n] = size(X);
Xloc = [];
Yloc = [];
eorder = [];
for i = 1:m
```

```
    ist = 0;
    for j = 1:n
        if(X(i,j) < range(2,j) && X(i,j) > range(1,j))
            ist = ist + 1;
        end
    end
    if(ist == n)
        Xloc(i,j) = X(i,j);
        Yloc(i) = Y(i);
        eorder(i) = i;
    end
end
Yloc = Yloc';
if (any(eorder))
    eorder(eorder == 0) =[];
    Xloc = Xloc(eorder, :);
    Yloc = Yloc(eorder);
end

% 模糊聚类
function result = FCMclust(data,param)
%data normalization
X=data.X;

f0=param.c;
%checking the parameters given
%default parameters
if exist('param.m')==1, m = param.m;else m = 2;end;
if exist('param.e')==1, e = param.m;else e = 1e-4;end;

[N,n] = size(X);
[Nf0,nf0] = size(f0);
X1 = ones(N,1);

% Initialize fuzzy partition matrix
rand('state',0)
if max(Nf0,nf0) == 1,       % only number of cluster given
  c = f0;
```

```
  mm = mean(X);            %mean of the data (1,n)
  aa = max(abs(X - ones(N,1)*mm)); %
  v = 2*(ones(c,1)*aa).*(rand(c,n)-0.5) + ones(c,1)*mm;
  for j = 1 : c,
    xv = X - X1*v(j,:);
    d(:,j) = sum((xv*eye(n).*xv),2);
  end;
  d = (d+1e-10).^(-1/(m-1));
  f0 = (d ./ (sum(d,2)*ones(1,c)));

else
  c = size(f0,2);
  fm = f0.^m; sumf = sum(fm);
  v = (fm'*X)./(sumf'*ones(1,n)); %
end;

f = zeros(N,c);              % partition matrix
iter = 0;                    % iteration counter

% Iterate
while  max(max(f0-f)) > e
  iter = iter + 1;
  f = f0;
  % Calculate centers
  fm = f.^m;
  sumf = sum(fm);
  v = (fm'*X)./(sumf'*ones(1,n));
  for j = 1 : c,
    xv = X - X1*v(j,:);
    d(:,j) = sum((xv*eye(n).*xv),2);
  end;
  distout=sqrt(d);
  J(iter) = sum(sum(f0.*d));
  % Update f0
  d = (d+1e-10).^(-1/(m-1));
  f0 = (d ./ (sum(d,2)*ones(1,c)));
end
```

```
fm = f.^m;
sumf = sum(fm);

%results
result.data.f=f0;
result.data.d=distout;
result.cluster.v=v;
result.iter = iter;
result.cost = J;
```

参 考 文 献

[1] Fu M. Optimization via simulation: a review. Annals of Operations Research, 1994, 53 (1/4) : 199-247.

[2] Boesel J, Bowden Jr. R. Glover F, et al. Future of simulation optimization. Proc Winter Simulation Conference, Arlington, 2001: 1466-1469.

[3] A zadivar F. Simulation optimization methodologies. Proc Winter Simulation Conference, Phoenix, 1999: 93-100.

[4] 余俊，周济. 优化方法程序库 OPB-2—原理及应用. 第一版. 武汉：华中理工大学出版社，1997.

[5] Songqing Shan, G. Gary Wang. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. Struct Multidisc Optim, 2010, 41: 219-241.

[6] Koch PN, Simpson TW, Allen JK, Mistree F. Statistical approximations for multidisciplinary design optimization: the problem of size. J Aircr 1999,36(1): 275-286.

[7] Gu L. A comparison of polynomial based regression models in vehicle safety analysis. Proceedings of 2001 ASME design engineering technical conferences—design automation conference, Pittsburgh, PA, 2001, 9: 9-12.

[8] Alexandrov N, Alter SJ, Atkins HL, Bey KS, Bibb KL, Biedron RT. Opportunities for breakthroughs in large-scale computational simulation and design: NASA/TM-2002-211747.

[9] Papalambros PY, Michelena NF. Model-based partitioning in optimal design of large engineering systems. Multidisciplinary design optimization: state-of-the-art. SIAM, 1997:209-226.

[10] Papalambros PY, Michelena NF. Trends and challenges in system design optimization. In: Proceedings of the international workshop on multidisciplinary design optimization, Pretoria, S. Africa, 2000, 8: 7-10.

[11] Dong-Hoon Choi, Hong-Soo Park, Min-Soo Kim. A direct treatment of min-max dynamic response optimization problems. AIAA-93-1352-CP.

[12] Chung-Ho Wang. Dynamic multi-response optimization using principal component analysis and multiple criteria evaluation of the grey relation model. Int J Adv Manuf Technol , 2007, 32: 617-624.

[13] J. Bausa, G. Tsatsaronis. Dynamic Optimization of Startup and Load-Increasing Processe.

[14] Arvind U. Raghunathan, Dharmashankar Subramanian, Lorenz T. Biegler, Tariq Samad.

Dynamic Optimization Strategies for Three-Dimensional Conflict Resolution of Multiple Aircraft. Journal of guidance, control, and dynamics. , 2004, 27(4): 586-594.

[15] Park S, Kapania RK, Kim SJ. Nonlinear transient responseand second-order sensitivity using time finite element method. AIAA J, 1999, 37(5):613-622.

[16] A. T. Patera. A spectral element method for fluid dynamics; laminar flow in a channel expansion. Journal of Computational Physics, 1984, 54: 468-488.

[17] M. H. Kurdi and P. S. Beran. Spectral element method in time for rapidly actuated systems. Journal of Computational Physics, 2007(3):1809-1835.

[18] Kang B-S, Park G-J, Arora JS. A review of optimization of structures subjected to transient loads. Struct Multidisc Optim, 2006, 31(2):81-95.

[19] Choi W S, Park GJ. Structual optimization using equivalent static loads at all time intervals. Comput Methods Appl Mech Eng, 2002, 191:2077-2094.

[20] Grandhi RV, Haftka RT, Watson LT. Design oriented identification of critical times in transient response. AIAA Pap, 1984: 1984-0899.

[21] G. Gary Wang and S. Shan. Review of Metamodeling Techniques in Support of Engineering Design Optimization. J. Mech. Des. 2007, 129(4): 370-380.

[22] Jin, R. , Chen, W. and Sudjianto, A. An Efficient Algorithm for Constructing Optimal Design of Computer Experiments. Journal of Statistical Planning and Inferences, 2005, 134(1): 268-28.

[23] Li Chen, Zhendong Ding, and Simon Li. Tree-based dependency analysis in decomposition and re-decomposition of complex design problems. ASME J Mech Des. 2005, 127: 12-23.

[24] Altus SS, Kroo IM, Gage PJ. A genetic algorithm for scheduling and decomposition of multidisciplinary design problems. ASME J Mech Des, 1996, 118: 486-489.

[25] Kusiak A, Larson N. Decomposition and representation methods in mechanical design. ASME J Mech Des, 1995, 117: 17-24.

[26] Li S. Matrix-based decomposition algorithms for engineering application: survey and generic framework. Int J Prod Dev, 2009, 9: 78-110.

[27] Michelena NF, Papalambros PY. A hypergraph framework for optimal model-based decomposition of design problems. Comput Optim Appl, 1997, 8(2):173-196.

[28] Michelena NF, Papalambros PY. A network reliability approach to optimal decomposition of design problems. ASME J Mech Des, 1995, 117:433-440.

[29] K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. Journal of the Association for Computing Machinery, 1995, 42:488-499.

[30] Wagner TC, Papalambros PY. A general framework for decomposition analysis in optimal design. Adv Des Autom, 1993, 2: 315-325.

[31] Chen L, Ding Z, Li S. A formal two-phase method for decomposition of complex design problems. ASME J Mech Des, 2005, 127:184-195.

[32] Suh NP. Axiomatic design: advances and applications. Oxford University Press, New York, 2001.

[33] K. J. Craig, Nielen Stander, D. A. Dooge and S. Varadappa. Automotive crashworthiness design using response surface-based variable screening and optimization. Engineering Computations: International Journal for Computer-Aided Engineering and Software, 2005, 22(1): 38-61.

[34] Myers, R. H. and Montgomery, D. C. Response Surface Methodology – Process and Product Optimization Using Designed Experiments, 2nd ed. , Wiley, New York, NY, 2002.

[35] Peter Craven and Grace Wahba. Smoothing Noisy Data with Spline Functions Estimating the Correct Degree of Smoothing by the Method of Generalized Cross-Validation. Numer. Math, 1979, 31: 377-403.

[36] Wang, G. G. and Simpson, T. W. , Fuzzy Clustering Based Hierarchical Metamodeling for Space Reduction and Design Optimization, Engineering Optimization, 2004, 36(3): 313-335.

[37] Ismail Bin Mohd. Identification of region of attraction for global optimization problem using interval symmetric operator. Applied Mathematics and Computation 2000, 110: 121-131.

[38] Reddy, S. Y. A Methodology for Early-Stage Exploration of Design Space. The proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, Paper No. 96-DETC/DAC1089, August 18-22, Irvine, California, 1996.

[39] Koch, P. N. , Simpson, T. W. , Allen, J. K. , Mistree, F. Statistical Approximations for Multidisciplinary Optimization: The Problem of Size. Special Multidisciplinary Design Optimization Issue of Journal of Aircraft, 1999, 36(1), 275-286.

[40] Vassili Toropov. Fred van Keulen. Valery Markine. Henk de Boer. Refinements in the multi-point approximation method to reduce the effects of noisy structural responses. AIAA Meeting Papers on Disc, 1996, 941-951.

[41] J. F. Rodriguez, J. E. Renaud, L. T. Watson. Trust Region Augmented Lagrangian Methods for Sequential Response Surface Approximation and Optimization. Transactions of the ASME, 1998, 120(3): 58-66.

[42] J. F. Rodriguez, J. E. Renaud. Convergence of trust using variable fidelity region augmented Lagrangian approximation data. Structural Optimization, 1998, 15: 141-156.

[43] Younis, A. A. , Xu, R. , and Dong , Z. , Approximated unimodal Region elimination based global optimization method for engineering design. International Journal of Product Development, 2009, 9(1/2/3): 164-187.

[44] Vinícius V. de Melo , Alexandre C. B. Delbem , Dorival L. P. Júnior , Fernando M. Federson . Improving global numerical optimization using a search-space reduction algorithm. Genetic And Evolutionary Computation Conference Proceedings of the 9th annual conference on Genetic and evolutionary computation, London, England, 2007.

[45] 吴义忠，吴民峰，陈立平. 基于 Modelica 语言的复杂机械系统统一建模平台研究[J]. 中国机械工程，2006, 17(22)：2391-2396.

[46] 赵建军，丁建完，周凡利，陈立平. Modelica 语言及其多领域统一建模与仿真机理[J]. 系统仿真学报，2006, 18(2): 570-573.

[47] 丁建完，陈立平，周凡利. 复杂陈述式仿真模型的指标分析. 系统仿真学报，2006,18(8)：2092-2096.

[48] 余俊，周济. 优化方法程序库 OPB-2—原理及应用. 第一版. 武汉：华中理工大学出版社，1997.

[49] K. A. High, R. D. LaRoche. Parallel nonlinear optimization techniques for chemical process design problems. Computers & Chemical Engineering, 1995, 19(6-7): 807-825.

[50] Mao Hu-ping, Wu Yi-zhong, Chen Li-ping. Multivariate Adaptive Regression Splines Based Optimization using Move Limit Strategy[J]. Journal of Shanghai University (English Edition), 2011, 15(6): 542-547.

[51] 张帆，邵之江，仲卫涛. 化工过程系统优化的分布式并行计算. 化工学报，2001，52（5）：396-400.

[52] 陈忠. 关于非线性规划问题的并行算法. 长江大学学报（自科），2006, 4(12)：1-8.

[53] Ling Jiang, Lorenz T. Biegler, V. Grant Fox. Design and optimization of pressure swing adsorption systems with parallel implementation. Computers and Chemical Engineering 2005(29): 393-399.

[54] Wilkinson,B. ,Allen,M. 并行程序设计（第二版）. 陈鑫达译. 北京：机械工业出版社，2005.

[55] 黄金贵，陈建二，陈松乔. 并行环境下基于多处理机任务的调度模型与调度算法[J]. 计算机科学，2002，29(4):1-3.

[56] Hochbaum,D. S. ed. Approximation algorithms for NP-hard problems. London : PWS Publishing Company,1997.

[57] 柴山，王健，曹新忠. 离散变量优化设计的方向差商法. 计算结构力学及其应用，1994,11(3)：283-293.

[58] 毛虎平，吴义忠，陈立平. 基于多领域仿真的 SQP 并行优化算法[J]. 中国机械工程，2009,20(15)：1823-1829.

[59] POZRIKIDIS C. Introduction to Finite and Spectral Element Methods Using Matlab, Chapman and Hall/CRC, 2005.

[60] PARTER S V. On the Legendre-Gauss-Lobatto Points and Weights. Journal of Scientific Computing, 1999, 14 (4):347-355.

[61] 毛虎平，苏铁熊，李建军，等. 基于逐步时间谱元法的结构动态响应仿真[J]. 中北大学学报（自然科学版），2013, 34(4)：424-430.

[62] Qian Wang, Jasbir S. Arora. Alternative Formulations for Transient Dynamic Response Optimization. 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference 18 – 21, Austin, Texas, 2005, 8.

[63] HAUG E J, ARORA J S, Applied optimal design: mechanical and structural systems, New York,

John Wiley & Sons, Inc, 1979.

[64] 薛定宇，陈阳泉. 高等应用数学问题的 MATLAB 求解. 北京：清华大学出版社，2004.

[65] Simpson T W, Peplinski J D, Koch P N et al. Metamodels for Computer-based Engineering Design: Survey and Recommendations, Engineering with Computers, 2001(17): 129-150.

[66] HSIEH C C AND ARORA J S. Design Sensitivity Analysis and Optimization of Dynamic Response, Computer Methods in Applied Mechanics and Engineering, 1984, 43: 195-219.

[67] HSIEH C C AND ARORA J S. A Hybrid Formulation for Treatment of Point-Wise State Variable Constraints in Dynamic Response Optimization [J]. Computer Methods in Applied Mechanics and Engineering, 1985, 48: 171-189.

[68] PAENG J K AND ARORA J S. Dynamic Response Optimization of Mechanical System with Multiplier Methods, ASME Journal of Mechanism, Transmission, and Automation in Design, 1989, 111: 73-80.

[69] CHOI D H, PARK H S, KIM M S. A direct treatment of min-max dynamic response optimization problems. AIAA-93-1352-CP.

[70] KURDI M H. AND BERAN P S Optimization of Dynamic Response using Temporal Spectral Element Method. 46th AIAA Aerospace Sciences Meeting and Exhibit 7 - 10 January 2008, Reno, Nevada.

[71] 毛虎平，吴义忠，陈立平. 基于时间谱元法的动态响应优化[J]. 机械工程学报，2010，46(16)：79-87.

[72] Afimiwala K A，Mayne R W. Optimal Design of an Impact Absorber. Journal of Engineering for Industry [J]. Journal of Engineering for Industry, 1974, 96(1): 124-130.

[73] Kocer F Y, Arora J S. Optimal design of latticed towers subjected to earthquake loading [J]. Journal of Structural Engineering, 2002, 128(2):197-204.

[74] Oral S, Ider S K. Optimum design of high-speed flexible robotic arms with dynamic behavior constraints [J]. Computers & Structures, 1997,65(2): 255-259.

[75] Grandhi R V, Haftka R T, Watson L T. Design-oriented identification of critical times in transient response [J]. AIAA Journal, 1986,24(4): 649-656.

[76] Hsieh C C, Arora J S. An efficient method for dynamic response optimization [J]. AIAA Journal, 1985, 23:1484-1486.

[77] Park K J, Lee J N, Park G J. Structural shape optimization using equivalent static loads transformed from dynamic loads [J]. International Journal for Numerical Methods in Engineering, 2005,63(4): 509-602.

[78] 秦仙蓉，廖鑫，李永凤，等. 7500t 浮式起重机金属结构的动态优化设计[J]. 中国机械工程，2012,10(2)：161-165.

[79] 马斌. 柴油机连杆的动态应力分析及优化设计[J]. 机械设计，2012,29(4)：59-62.

[80] 杜雪松，朱才朝，宁杰. 船用齿轮传动的动态优化设计[J]. 重庆大学学报，2011,34(5)：14-18.

[81] 王晓博，谢瑞清，丁武学，等. 锤片式粉碎机转子结构动态优化设计[J]. 振动与冲击，2010,29(5)：147-161.

[82] 乔崇全，王克杰. 大型振动筛动力学分析及动态优化设计[J]. 煤矿机械，2011,32(11)：3-5.

[83] 杨斌，张其方，许锋，等. 基于动态响应的变约束迭代优化及其在化工过程操作优化中的应用[J]. 计算机与应用化学，2011,28(12)：1521-1526.

[84] 王志强，邵之江，万娇娜，等. 基于径向基函数的动态优化问题联立求解方法[J]. 上海交通大学学报，2011,45(8)：1221-1225.

[85] 史洪岩，苑明哲，王天然，等. 间歇过程动态优化方法综述[J]. 信息与控制，2012,41(1)：76-82.

[86] 毛虎平，吴义忠，陈立平. 基于多领域仿真的 SQP 并行优化算法[J]. 中国机械工程，2009,20(15)：1823-1829.

[87] DONALD R. JONES, MATTHIAS SCHONLAU, WILLIAM J. WELCH. : Efficient Global Optimization of Expensive Black-Box functions. Journal of Global Optimization, 1998, 13: 455-492.

[88] Friedman J H. Multivariate Adaptive Regression Splines, The Annals of Statistics, 1991, 19(1): 1-67.

[89] 邹林君，吴义忠，毛虎平，等. Kriging 模型的增量构造及其在全局优化中的应用[J]. 计算机辅助设计与图形学学报，2011，23(4)：1-7.

[90] WANG L Q, SHAN S, WANG G G. Mode-pursuing sampling method for global optimization on expensive black-box functions [J]. Engineering Optimization, 2004, 36(4)：419-438.

[91] 顾纪超，李光耀，干年妃，等. 基于混合元模型的准最优策略在汽车轻量化中的应用[J]. 中国机械工程，2011,22(16): 1997-2000.

[92] Mao H P, Wu Y Z, Chen L P. Data Driven Multivariate Adaptive Regression Splines Based Simulation Optimization [J]. Applied Mechanics and Materials, 2011, 44: 3800-3806.

[93] 毛虎平，苏铁熊，李建军. 多元模型自适应与时间谱元法结合的动态优化[J]. 计算机辅助设计与图形学学报，2013, 65(6): 388-390.

[94] 毛虎平，吴义忠，李建军. 时间谱元法在动态响应优化中的应用[J]. 振动工程学报，2013, 26(3)：395-403.

[95] Gu, L. A Comparison of Polynomial Based Regression Models in Vehicle Safety Analysis, ASME Design Engineering Technical Conferences - Design Automation Conference (DAC) (Diaz, A., ed.), Pittsburgh, PA, ASME, September 9-12, 2001, Paper No. DETC2001/DAC-21063.

[96] L. A. Schmit Jr. , B. Farshi, Some approximation concepts for structural synthesis. AIAA J. 1974, 12 (5): 692-699.

[97] L. A. Schmit, Jr. , H. Miura, Approximation concepts for efficient structural synthesis, Tech. Report CR-2552, NASA, 1976.

[98] Box, G. E. P. , and Wilson, K. B. On the Experimental Attainment of Optimum Conditions, Journal of the Roal Statistical Society. Series B(Methodological), 1951, 13(1): 1-45.

[99] Myers, R. H. and Montgomery, D. Response Surface Methodology: Process and Product Optimization Using Designed Experiments, John Wiley and Sons, Inc. , Toronto, 2002.

[100] Major Scott T. Crino. Combining MARS with RSM for Simulation-based Design Optimization: [Ph. D. Thesis]. University of Virginia, 2006.

[101] Friedman, J. H. Multivariate Adaptive Regression Splines, The Annals of Statistics, 1991, 19(1): 1-67.

[102] Vassili Toropov. Fred van Keulen. Valery Markine. Henk de Boer. Refinements in the multi-point approximation method to reduce the effects of noisy structural responses AIAA Meeting Papers on Disc, 1996, 941-951.

[103] Montgomery, D. Design and Analysis of Experiments. John Wiley and Sons, New York, 1991.

[104] Taguch G. , Yokoyama Y, Wu Y. Taguchi Methods: Design of Experiments. Amer Supplier Inst, Michigan,1993.

[105] Richardson S, Wang S, Jennings L S. A multivariate adaptive regression B-spline algorithm (BMARS) for solving a class of nonlinear optimal feedback control problems. Automatica, 2008, 44(4): 1149-1155.

[106] Loh W L. On Latin Hypercube Sampling. The Annals of Statistics, 1996, 24(5): 2058-2080.

[107] McKay M D, Beckman R J, Conover W J. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. Technometrics, 1979, 21(2):239-245.

[108] 张润楚，王兆军，关于计算机试验的设计理论和数据分析，应用概率统计，1994,10(4)：420-435.

[109] Rodriguez J. F. , Renaud J. E. , Watson L. T. Trust Region Augmented Lagrangian Methods for Sequential Response Surface Approximation and Optimization. Transactions of the ASME. 1998, 120(3): 58-66.

[110] 邢静忠，王永刚，陈晓霞，等. ANSYS7. 0 分析实例与工程应用. 北京：机械工业出版社，2004.

[111] Schmit, L. A. , Jr. Farshi, B. Some approximation concepts for structural synthesis. AIAA J. 1974, 12: 692-699.

[112] Schmit, L. A. , Miura, H. Approximation concepts for efficient structural synthesis. NASA CR-2552, 1976.

[113] S. Shan and G. G. Wang. Space exploration and global optimization for computationally intensive design problems: a rough set based approach. Struct Multidisc Optim, 2004, 28: 427-441.

[114] Sacks J, Welch W J, Mitchell T J, et al. Design and Analysis of computer experiments.

Statistical. Science. 1989, 4(4): 409-423.

[115] Søren N. Lophaven, Hans Bruun Nielsen, Jacob Søndergaard. DACE—A MATLAB Kriging Toolbox. Technical University of Denmark, 2002.

[116] Sasena M J．Flexibility and efficiency anhancements for constrained global design optimization with kriging approximations: [PhD thesis]．University of Michigan，2002.

[117] Jay D. Martin, Computational Improvements to Estimating Kriging Metamodel Parameters. Journal of Mechanical Design, 2009, 131(8): 084501-084506.

[118] Søren N. Lophaven, Hans Bruun Nielsen, Jacob Søndergaard. ASPECTS OF THE MATLAB TOOLBOX DACE. Technical University of Denmark. 2002.